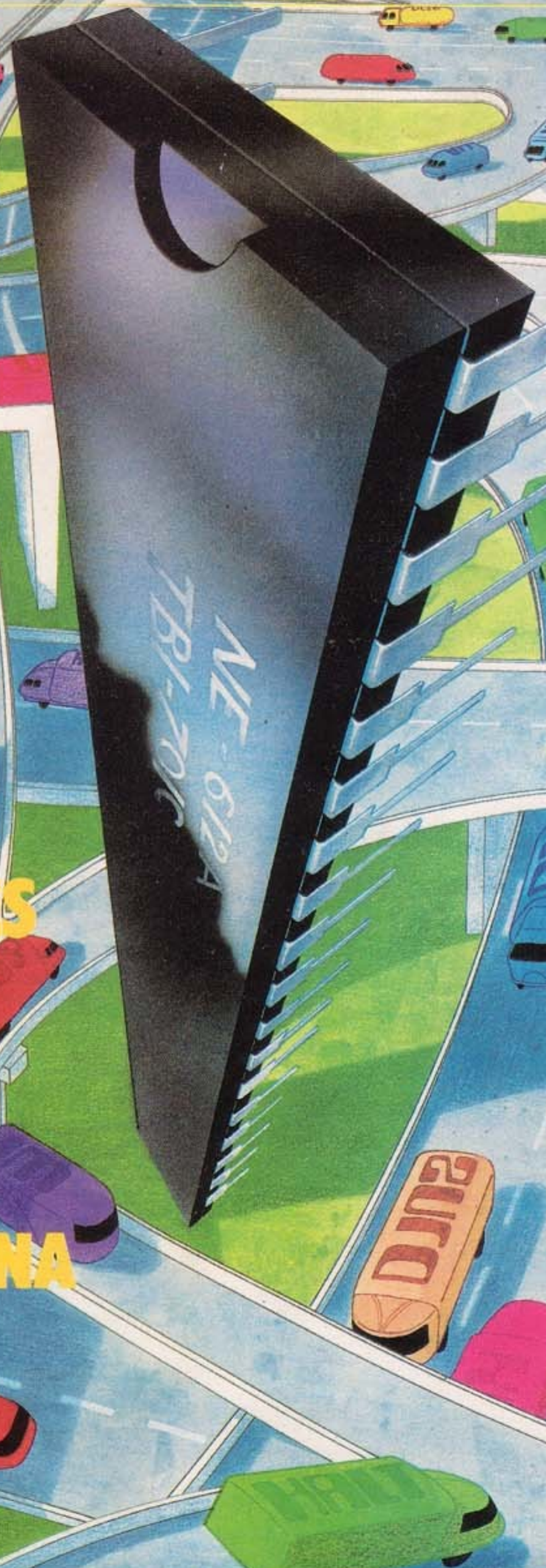
INPUT

Publicación práctica para usuarios de

sinclair

Revista mensual 1986 Precio 350 Ptas Año 2 Número 12



TODO SOBRE READ Y DATA

MANEJO DE INTERRUPCIONES

HARDCOPY EN CODIGO MAQUINA

Regálate lo último en Vídeo/Juegos

GRATIS
UNA BOLSA PETATE
por la compra de los 2 juegos

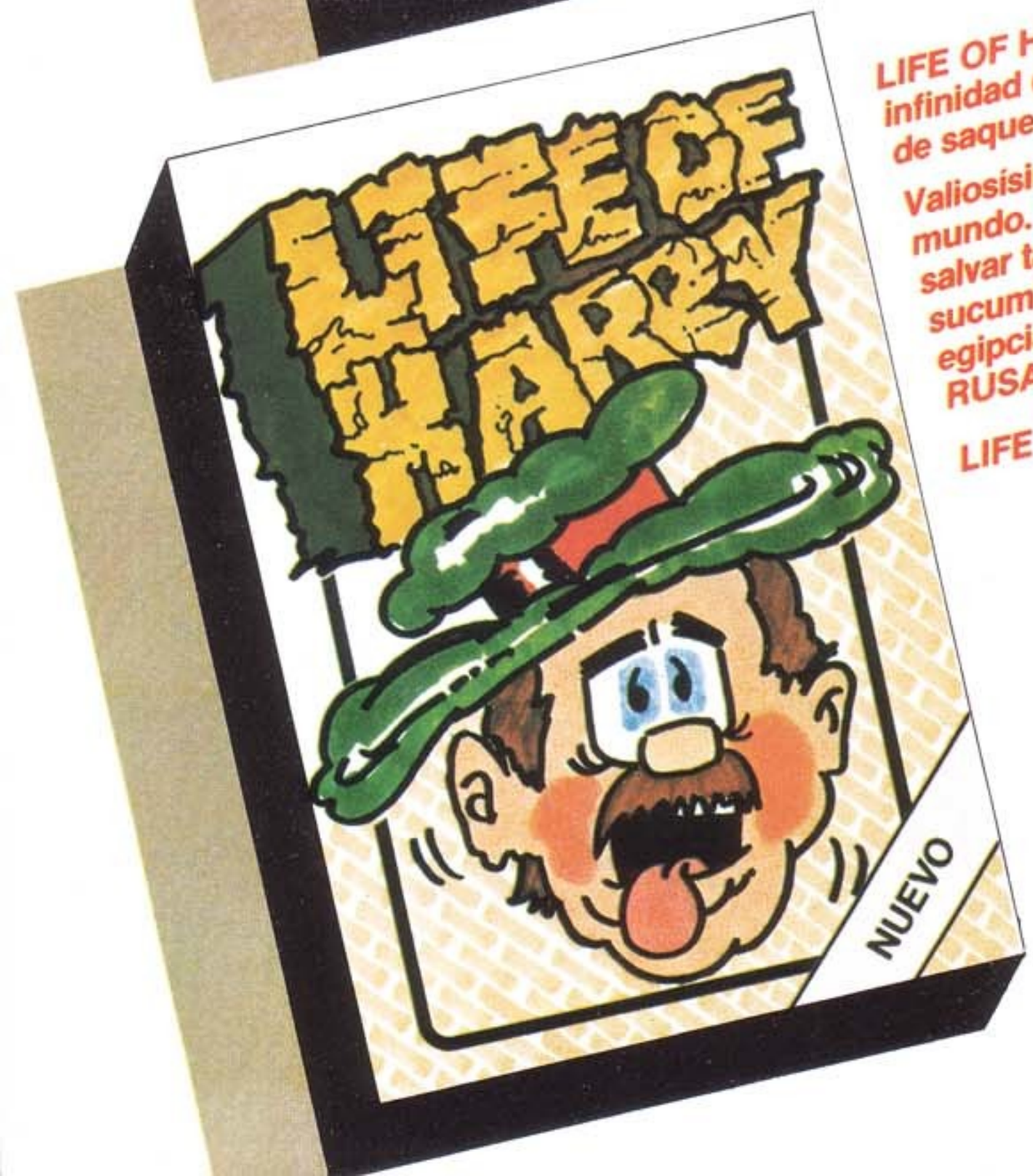
El combate aéreo ACE es por el momento el vuelo simulado más realista que puedes conseguir.

Por primera vez en un ordenador doméstico sientes que es realmente como volar en una operación aérea militar. Atacando tanques, barcos, helicópteros, lanzando misiles y evitando el inteligente juego aéreo del enemigo.

Eliges: Verano, invierno, vuelo nocturno, misión sencilla o doble y varios niveles de Juego. Arma: Ataque tierra, ataque naval, mapa del satélite, despegue, aterrizaje, repostar en el aire, radar, etc.

Con más de 300 imágenes diferentes ACE tiene las vistas más detalladas de árboles, colinas, y todos los objetos del suelo, incluyendo sus magníficas vistas en tres dimensiones.

AIR COMBAT EMULATOR: 2.600 Pts.



LIFE OF HARRY: juego de aventuras, donde podrás recorrer infinidad de lugares del mundo y ayudar a HARRY a salvar al mundo de saqueadores.

Valiosísimos tesoros están desapareciendo de todos los lugares del mundo. Harry ha sido designado por el Gobierno Británico para salvar tantos artículos como sea posible antes de que estos sucumban por el placer de los saqueadores. Desde las tumbas egipcias a los bares más olvidados de Australia, desde las minas RUSAS hasta la NASA.

LIFE OF HARRY: 2.400 Pts.

Sí, remítame contra reembolso el/los juegos que a continuación señalo con una "X"

AIR COMBAT EMULATOR, SPECTRUM ☐ COMMODORE ☐
LIFE OF HARRY, SPECTRUM ☐
AMPLIACION DE MEMORIA (3.200 Pts.) ☐
INTERFACE (1.500 Pts.) ☐

NOMBRE .....

CALLE O PLAZA ..... Nº .....

POBLACION .....

PROVINCIA ..... C. P .....

slyco

Pídelo en tu tienda habitual o recorta el cupón adjunto y envíalo a SLYCO, S.A. Teléfonos: 413 74 65/66
Corazón de María, 60, 4ºA2 - 28002 MADRID

AÑO 1 NUMERO 12

DIRECTOR:
Alejandro Diges
COORDINADOR EDITORIAL:
Francisco de Molina
DISEÑO GRAFICO:
Tomás López
COLABORADORES:
Antonio Taratiel, Luis R. Palencia, Francisco Tórtola, Benito Román, Esther de la Cal, Ernesto del Valle, Equipo Molisoft.

INPUT Sinclair es una publicación juvenil de EDICIONES FORUM

GERENTE DIVISION DE REVISTAS:
Angel Sabat

PUBLICIDAD: José Real-Grupo Jota
Madrid: c/ General Varela, 35
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.º
Teléf. 250 23 99

FOTOMECANICA: Ochoa, S. A.

COMPOSICION: EFCA, S. A.

IMPRESION: Sirven Grafic
C/ Gran Via, 754-756. 08013 Barcelona
Depósito legal: B-21954-1986

SUSCRIPCIONES: EDISA,
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12

REDACCION:
Paseo de la Castellana, 93, 14.ª
28046 Madrid. Teléf. 456 54 13

DISTRIBUIDORA
R.B.A. PROMOTORA DE EDICIONES, S. A.
Travesera de Gracia, 56. Edificio Odiseus.
08006 Barcelona.

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

Se ha solicitado el control OJD

INPUT sinclair

SUMARIO

EL PRIMER AÑO

Con éste son ya doce números en la calle; mejor dicho: trece. El especial de verano fue añadido como consecuencia de vuestra inestimable aceptación. Igualmente tuvimos en cuenta el éxito del cuadernillo de programas que incluíamos en el ejemplar del pasado diciembre. De ahí salió un número diferente que potencia la fiel continuidad de la cita mensual.

El balance de estos doce meses nos hace darnos cuenta de vuestro alto índice de participación en las actividades a las que os hemos convocado, desde concursos, votaciones..., sin olvidarnos de la abrumadora respuesta a la encuesta.

Hemos publicado unas cuantas colaboraciones de lectores cuyo excelente nivel las hizo pasar desapercibidas (para bien) entre los artículos desarrollados por autores con experiencia. Sin embargo, la cantidad de cintas recibidas con programas vuestros, no siendo pocas, distan algo de nuestras expectativas. Os animamos a enviarlos, un buen programa no puede dormitar en un cajón.

Por otro lado, durante tan corto período, hay que ver cómo ha evolucionado este mundo nuestro, con la aparición de nuevas marcas y modelos de ordenador más competitivos y, sobre todo, lo que es casi increíble ha sido la evolución de los precios a la baja.

También es cierto que se han malogrado algunos otros sistemas, en los que se habían puesto grandes esperanzas por su aparente competitividad y todos conocemos cuáles son.

Este número vuelve a seguir la línea de los once anteriores, que se caracteriza por su intención didáctica frente a la opción de publicar fundamentalmente listados para copiar, sin rehuír suministrar algunos de vez en cuando.

Desde este editorial os volvemos a instar para que continuéis enviando vuestras colaboraciones y programas. Estamos seguros de que habéis preparado trabajos sensacionales durante el estío.

LOS MEJORES DE INPUT

Hemos pensado que es interesante disponer de un ***ranking*** que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en **INPUT**.

El resultado de las votaciones será publicado en cada número de **INPUT**.

Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.

Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: **LOS MEJORES DE INPUT** Alberto Alcocer, 46 - 4.º B. 28016 Madrid

ELIGE TUS PROGRAMAS

Primer título elegido

Segundo título elegido

Tercer título elegido

Programa que te gustaría conseguir

Qué ordenador tienes

Nombre

1.er Apellido

2.º Apellido

Fecha de nacimiento

Teléfono

Dirección

Localidad

Provincia

LA NUEVA VERSION DEL SPECTRUM

La esperada remodelación del Spectrum parece que será bautizada como Spectrum Plus Two. Todo el mundo supone que su puesta de largo para principios del presente mes de septiembre, coincidiendo con el PCW Show que se celebra en Londres.
Se especula en torno al tipo de teclado que incorporará el nuevo modelo, casi todos sospechaban que sería el mismo que utilizan los modelos de Amstrad. Sin embargo parece que se mantendrá el teclado del Plus, porque el cambio lo impide la razón de los costos.
Es muy posible que el nuevo diseno incorpore el ansiado *port* destinado al *joystick*, que tanto se ha echado de menos en los modelos actuales.
En cuanto al diseno interior del Plus Two, ahora pasa a disponer de cuatro ULAs de aplicación especializada, desarrolladas a la medida del sistema.
Por último, el grabador/reproductor de *cassette* se espera que sea el mismo que utilizan los otros modelos de Amstrad.
El precio anunciado al que podría ser lanzado en las islas británicas es de casi 140 Libras esterlinas, realmente competitivo en precio frente a los sistemas de Amstrad.
Lo cierto es que algunas importantes casas de *software* llevan un tiempo dedicadas a la misión casi secreta de dotar al nuevo 128 de un amplio soporte de programas que lo hagan atractivo para el usuario.
La intención comercial parece clara, se trataría de presentar al nuevo Sinclair como una excelente máquina de juegos con muchas posibilidades y bajo precio.
Nada se habla en torno a la compatibilidad con el *software* ya existente, pero parece asegurada.

LO NUEVO DE BEYOND

La firma Beyond está a punto de lanzar un programa en torno al que se han creado muchas espectativas. Lleva por titulo Dark Sceptre y presenta personajes animados de gran tamaño, en la linea que en su momento iniciara Tir Na Nog.
Se trata de un juego de aventuras recreado en la Edad media y el escenario es una isla en la que debes captar soldados para un ejercito que te ayudará a derrotar a los senores de las sombras
La perfección y grado de detalle con que están realizados los fondos y, particularmente, los personajes le convierten en un previsible éxito.

PROGRAMA DE SIGUE SIGUE SPUTNIK

El conjunto de moda Sigue Sigue Sputnik afirma haber desarrollado un programa de juego y están buscando quien se lo edite. El titulo inicialmente barajado es F-i-ii, Love missile.
La fecha que mas les gusta para el lanzamiento habría de coincidir con el lanzamiento de su nuevo disco "El chico del siglo 21".

DAVID BOWIE TAMBIEN ENTRA EN EL *SOFTWARE*

Otro conocido cantante se ha decidido también a participar en el mundo de los programas. Esta vez es David Bowie quien entra en los planes de la firma Activision, que ha obtenido los derechos para desarrollar un juego de aventura fantástica siguiendo la trama de la pelicula Laberinto
El papel de Bowie consiste en dar vida al rey de los duendes. El argumento gira en torno a las fantasias de una nina que desea que los *gnomos* y las hadas se lleven a su hermanito, para asi no tener que cuidarlo.
El lanzamiento está previsto para finales de año, coincidiendo con el estreno de la pelicula.

PROGRAMA BASADO EN *SPLITTING IMAGES*

El conocido programa de la televisión británica que hace furor en nuestras pantallas, Splitting Images llega ahora al Spectrum de la mano de la casa de *software* Domark en un programa titulado Split Personalities.
En esencia el juego consiste en un *puzzle* con personajes conocidos, aderezado con algo de dificultad en el manejo para asegurar un mayor interés.
Otra novedad que tiene prevista esta firma es el popular juego Trivial Pursuit. Es de suponer que no sea desarrollada una versión en castellano, aunque podrá ser una buena ocasión para poner a prueba nuestros conocimientos del idioma inglés.

* * * *

MANEJO DEL BORDER

La rutina que presentamos a continuación consiste en un ejemplo de utilización práctica de la instrucción BORDER desde el Código Máquina, consiguiendo interesantes efectos de coloración a modo de franjas de diversos colores que el lector podrá elegir. Es una rutina «vistosa» y corta (sin que por ello deje de ser interesante).

Al manejo de BORDER en sí, se le han añadido otras facilidades que señalamos a continuación.

Desde Código Máquina se puede utilizar la rutina de varias maneras, al igual que en BASIC. La correcta y verdadera sería utilizando la rutina ubicada en la dirección de memoria 8859 de la ROM. Así, cargando en el registro A el valor del color elegido, y llamando a la rutina se consigue el efecto deseado:

```
LD   A,"color"
CALL 8859
```

La alternativa que tenemos es utilizar:

```
LD   A,"color"
OUT  (#FE),A
```

con lo que conseguimos un cambio «temporal» del color del borde, porque el nuevo valor no es almacenado en la variable del sistema ATTR P (23693), que conserva los colores «permanentes» de tinta, papel, borde, etc... En este caso el nuevo valor es almacenado en la variable del sistema ATTR T (23694) que conserva los colores «temporales». Es decir, si desde el BASIC ejecutas:

```
BORDER 0: OUT 254,2
```

conseguirás en principio el color rojo, pero si pulsas cualquier tecla volverás a obtener el negro (el permanente).

En nuestro programa utilizamos la segunda opción para conseguir la sincronización adecuada. Es esencial escoger el tiempo exacto con que cambia de color el borde, si es mayor o menor no se consigue el efecto de una manera tan clara. Además desde el Código Máquina es más difícil conseguir la citada sincronización.

En relación con este último punto hay que destacar el uso de los bucles de retardo introducidos en las líneas 100 a la 150. Su función es «perder» tiempo, contrarrestando la rapidez del Código Máquina.

La rutina es un bucle que se repite indefinidamente al igual que los realizados en BASIC, pero con la diferencia de no poder recuperar el control una vez iniciada su ejecución. Para paliar esta posible desventaja se han introducido las líneas 210 a 240. Su función consiste en explorar el teclado y devolver el control al usuario una vez detectada la pulsación de la tecla elegida.

Estas últimas líneas suponen un gran paso adelante en la programación en C.M., brindándonos la ocasión de ejercer un control más real de nuestros programas, al pasar del C.M. al BASIC con mayor facilidad.

El **Spectrum** lee el teclado cada 20 milisegundos, dividiendo las cuatro filas de teclas en 8 semifilas, que se controlan desde el Código Máquina utilizando el valor asignado a cada semifila, y dentro de cada semifila comprobando la posible activación o desactivación del bit adecuado con instrucciones como AND o BIT que reflejan su resultado en el *flag* Z. Los valores mencionados se indican en el cuadro de la izquierda.

En nuestro caso intentamos comprobar la pulsación ENTER :

```
LD  A,#BF    ;Codigo de la
              semifila de
              ENTER
```

BIT INDICADOR

0	1	2	3	4	
C.Shift	Z	X	C	V	#FE
A	S	D	F	G	#FD
Q	W	E	R	T	#FB
1	2	3	4	5	#F7
0	9	8	7	6	#EF
P	O	I	U	Y	#DF
Enter	L	K	J	H	#BF
Space	S.Shift	M	N	B	#7F

CODIGOS DE SEMIFILA

```
IN   A,(#FE) ;Comun para      AND 1       ;Tambien se puede
               todos los                   hacer con BIT
               casos                       0,A
```

Si la tecla no es pulsada, el valor de respuesta será del tipo XXX11111. Si se presiona, el byte de respuesta será del tipo XXX11110, el cual al sumarle 00000001 (con AND 1) activará el *flag* Z, dando pie a una bifurcación condicional del tipo JR Z o JR NZ (igual con JP Z o JP NZ).

RUTINA 1

```
10          ORG  60000   ;Lugar de colocacion del C/M
20          ENT  60000   ;Direccion de ejecucion
30 OTRO     LD   B,10    ;La rutina se hace 10 veces
40 BUCLE1   PUSH BC      ;Contador del no. de veces que se hace
50          LD   B,10    ;Para coger los 10 colores de la tabla
60          LD   IX,VAL  ;Se situa en la tabla de valores VAL
70 BUCLE2   PUSH BC      ;Contador del no. colores tomados
80          LD   A,(IX)  ;Guarda un valor de la tabla en A
90          OUT  (#FE),A ;Borde del color contenido en A
100         LD   B,0     ;De la linea 100 a la 150 inclusive
110 BUCLE3  DJNZ BUCLE3  ; constituyen los bucles de retardo
120         LD   B,0     ; cuya funcion especifica es perder
130 BUCLE4  DJNZ BUCLE4  ; el tiempo exacto para que se consiga
140         LD   B,11    ; el efecto deseado. Esto es debido a
150 BUCLE5  DJNZ BUCLE5  ; la rapidez del c/m
160         INC  IX      ;Se situa en el sig. valor en la tabla
170         POP  BC      ;Saca el valor introducido en 70 y lo
180         DJNZ BUCLE2  ;disminuye en 1, saltando si es <> 0
190         POP  BC      ;Saca el valor del contador del prog.
200         DJNZ BUCLE1  ;Lo disminuye en 1, y salta si es <> 0
210         LD   A,#BF   ;Las lineas 210,220,230 esperan a que
220         IN   A,(#FE) ; pulsemos la tecla ENTER. Una vez pul-
230         AND  1       ; sada se activa el flag Z, RETornando
240         JR   NZ,OTRO ; al BASIC. Si no se pulsa va a OTRO
250         RET
260 VAL     DEFB 0,7,6,2,0,4,2,3,6,1 ;Tabla de valores (colores)
```

Como resumen de todo lo expuesto, pretendemos simular la siguiente rutina:

```
10 PAUSE 1: BORDER 1: BORDER
   1: BORDER 6: BORDER 2:
   BORDER 0: BORDER 4:
   BORDER 2: BORDER 3:
   BORDER 7: BORDER 1:
   GO TO 10
```

El equivalente en Código Máquina se muestra en la RUTINA 1.

En la línea 260 se han introducido los valores elegidos para las franjas del borde, por lo que pueden variarse sin problemas. También, si deseamos conseguir un resultado más estático, se puede modificar el valor de la línea 140 por 12.

Para quienes no tengan ensamblador, adjuntamos el programa cargador de la rutina:

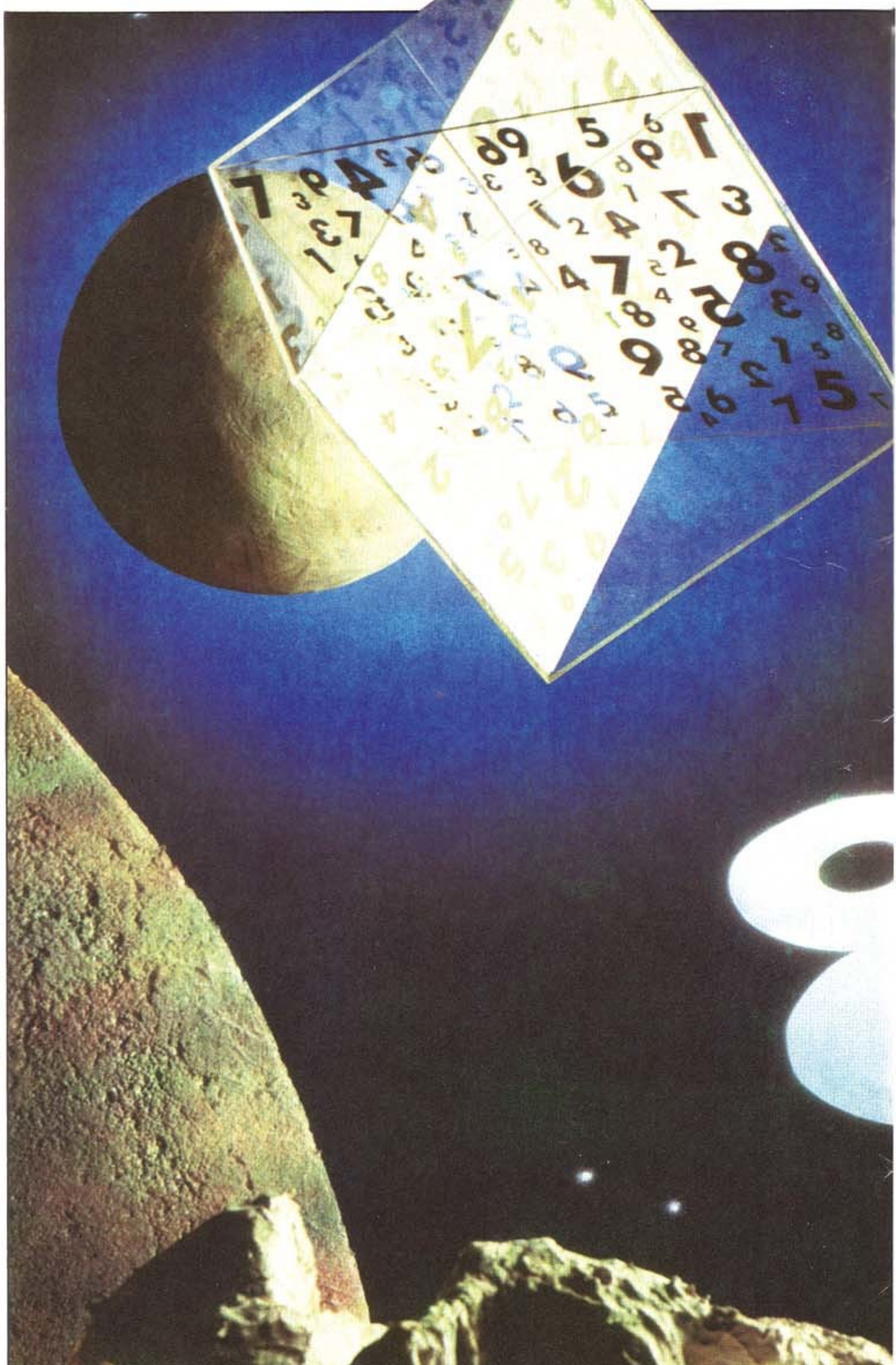
```
10 CLEAR 59999: FOR N=60000
   TO 60053: READ A: POKE N,
   A: NEXT N
20 DATA 6,10,197,6,10,221,33
   ,140,234,197,221,126,0,
   211,254,6,0,16,254,6,0,
   16,254,6,11,16,254,221,
   35,193,16,233,193,16,223
   ,62,191,219,254,230,1,32
   ,213,201,0,7,6,2,0,4,2,3
   ,6,1
```

Los 10 últimos valores en la lista de DATAs corresponden a los colores elegidos para la rutina, pudiéndose modificar. La rutina es reubicable.

Para ejecutar la rutina RANDOMIZE USR 60000.

MANEJO DE LA ARITMETICA HEXADECIMAL

Se puede aprender a contar con los dedos y aprender a contar en base 16. Pero aunque no tengas 16 dedos, encontrarás el manejo de cifras hexadecimales mucho más sencillo que tener que vértelas con grandes ristras de ceros y unos.



Los ordenadores, en lo más profundo de sus circuitos hacen toda su aritmética en binario, utilizando un sistema de números compuesto totalmente por ceros y unos, lo cual crea ciertas dificultades a los operadores humanos.

En cuanto los números alcanzan un tamaño moderado, en seguida te enfrentas con más roscos que una fábrica de *donuts*. Y no resulta sencillo teclear una serie larga de ceros y unos. Es muy fácil cometer un error y es muy difícil detectarlo después.

La manera que tiene el operador para evitar estas dificultades es utilizar una base de numeración diferente.

Los números hexadecimales son los que utilizan como base de numeración el número 16. Son lo suficientemente próximos a los números de base 10 como para resultar de manejo relativamente sencillo por parte de un operador humano.

Además 16 es igual a 2 × 2 × 2 × 2, lo cual significa que la conversión entre binario y hexadecimal es simple. El número decimal 16 se escribe 10 en hexadecimal y 10000 en binario. Además, todo número entre el 0 y el 15 se representa mediante un número binario de cuatro dígitos.

Para poder utilizar un sistema de numeración con una base mayor que 10, hay que definir símbolos para los nuevos dígitos.

En hexadecimal, 10 se representa por A, once por B, doce por C, trece por D, catorce por E y quince por F.

- POR QUE SE UTILIZAN LOS NUMEROS HEXADECIMALES
- CONTANDO DE 16 EN 16
- RELACION ENTRE BINARIO Y HEXADECIMAL

CONVERSION DE BINARIO A HEXADECIMAL

La conversión a hexadecimal de los números binarios de ocho dígitos que utilizan los ordenadores domésticos, es particularmente fácil. Tienes que dividir el número en dos secciones de ocho dígitos. A continuación, los primeros cuatro dígitos se convierten directamente en un dígito hexadecimal, y los cuatro restantes se convierten en otro dígito.

La conversión de decimal a hexadecimal es más difícil. Para hacerla tienes que hacer divisiones sucesivas del número decimal por 16. Los restos resultantes en cada división son los dígitos hexadecimales buscados.

Por ejemplo, al dividir 1226 por 16, obtienes como cociente 76 y de resto 10. En hexadecimal 10 se escribe A. 76 dividido por 16 da cuatro y de resto 12, que en hexadecimal se escribe C. Por último, 4 dividido entre 16 da 0 y de resto 4. En consecuencia 1226 en decimal se escribe 4CA en hexadecimal.

El siguiente programa es bastante largo, pero merece la pena teclearlo, porque te ayudará a asentar en tu mente la forma en que se opera esta conversión:

Teclea para Spectrum

```
20 CLS
25 PLOT 140,0: DRAW 0,160
30 PRINT  INVERSE 1;AT 0,8;
   " BIN,DEC,HEX "
40 PRINT INVERSE 1;AT 4,2;
   "[2*ESPACIO]BINARIO:
   [5*ESPACIO]"
50 PRINT INVERSE 1;AT 9,2;
   "[2*ESPACIO]DECIMAL:
   [4*ESPACIO]"
60 PRINT AT 10,5;
   "+[3*ESPACIO]+[3*ESPACIO]
   +[3*ESPACIO]+[3*ESPACIO]+
   +[3*ESPACIO]+[3*ESPACIO]+"
70 PRINT INVERSE 1;AT 17,2;
   "[2*ESPACIO]HEXADECIMAL:"
80 PRINT AT 18,4;
   "+[2*ESPACIO]+[2*ESPACIO]
   +[2*ESPACIO]="
90 PRINT AT 18,20;
   "+[2*ESPACIO]+[2*ESPACIO]
   +[2*ESPACIO]="
100 LET NO=0
110 GO TO 150
120 LET A$=INKEY$: IF A$=""
    THEN GO TO 120
130 IF A$=" " THEN  LET NO=
    NO+1: IF NO=256 THEN LET
    NO=0
135 IF A$="B" THEN LET NO=NO
    -1: IF NO=-1 THEN LET NO
    =255
140 IF A$="B" OR A$=" " THEN
    GO TO 150
145 INPUT "?";NO
150 GO SUB 170: GO SUB 250
160 GO TO 120
170 LET NU=NO: LET C=128
175 FOR X=0 TO 7
180 LET N=0: IF NU>=C THEN
    LET N=1: LET NU=NU-C
190 LET C=C/2
200 PRINT AT 5,2+4*X;N
210 IF N=1 THEN PRINT AT 10,
    2+4*X;C*2
220 IF N=0 THEN PRINT AT 10,
    2+4*X;"0[2*ESPACIO]"
230 NEXT X
235 PRINT AT 13,6;"TOTAL
    DECIMAL= ";NO;
    "[2*ESPACIO]"
240 RETURN
250 LET HI=INT (NO/16): LET
    HH=HI
260 LET LO=(NO-HI*16): LET
    LL=LO: IF LO>9 THEN LET
    LO=LO+7
265 IF HI>9 THEN LET HI=HI+7
270 LET HI=HI+48: LET LO=LO+
    48
280 PRINT AT 18,14;CHR$ HI;
    AT 18,30;CHR$ LO
290 LET C=8
300 FOR X=0 TO 3
310 LET N=0: IF HH>=C THEN
    LET N=C: LET HH=HH-C
315 LET M=0: IF LL>=C THEN
    LET M=C: LET LL=LL-C
320 LET C=C/2
330 PRINT AT 18,2+X*3;N;AT
    18,18+X*3;M
340 NEXT X
400 PRINT AT 21,6;"TOTAL HEX
    = ";CHR$ HI;CHR$ LO
500 RETURN
```

```
20 CLS
30 PRINT AT 0,9;
   " BIN,DEC,HEX "
40 PRINT AT 4,4;
   "[2*ESPACIO]BINARIO:
   [5*ESPACIO]"
50 PRINT AT 9,4;
   "[2*ESPACIO]DECIMAL:
   [4*ESPACIO]"
60 PRINT AT 10,5;
   "+[3*ESPACIO]+[3*ESPACIO]
   +[3*ESPACIO]+[3*ESPACIO]+
   [3*ESPACIO]+[3*ESPACIO]+"
70 PRINT AT 17,4;
   "[2*ESPACIO]HEXADECIMAL:"
80 PRINT AT 18,4;
   "+[2*ESPACIO]+[2*ESPACIO]
   +[2*ESPACIO]="
90 PRINT AT 18,20;
   "+[2*ESPACIO]+[2*ESPACIO]
   +[2*ESPACIO]="
100 LET NO=0
110 GO TO 150
120 LET A$=INKEY$
125 IF A$="" THEN  GOTO 120
130 IF A$=<>"F"THEN GOTO 135
131 LET NO=NO+1:
132 IF NO=256 THEN LET NO=0
135 IF A$<>"B" THEN GOTO 140
136 LET NO=NO-1
137 IF NO=-1 THEN LET NO=255
140 IF A$="B" OR A$="F" THEN
    GOTO 150
145 INPUT NO
150 GOSUB 170
155 GOSUB 250
160 GOTO 120
170 LET NU=NO
171 LET C=128
175 FOR X=0 TO 7
180 LET N=0
185 IF NU>=C THEN LET N=1
186 IF NU>C THEN LET NU=NU-C
190 LET C=C/2
200 PRINT AT 5,2+4*X;N
210 IF N=1 THEN PRINT AT 10,
    2+4*X;C*2
220 IF N=0 THEN PRINT AT 10,
    2+4*X;"0[2*ESPACIO]"
230 NEXT X
235 PRINT AT 13,6;"TOTAL
    DECIMAL= ";NO;
    "[2*ESPACIO]"
240 RETURN
250 LET HI=INT (NO/16)
255 LET HH=HI
260 LET LO=(NO-HI*16)
261 LET LL=LO
270 LET HI=HI+28
275 LET LO=LO+28
280 PRINT AT 18,14;CHR$ HI;
    AT 18,30;CHR$ LO
290 LET C=8
300 FOR X=0 TO 3
310 LET N=0
311 IF HH>=C THEN LET N=C
312 IF HH>=C THEN LET HH=
    HH-C
315 LET M=0
316 IF LL>=C THEN LET M=C
317 IF LL>=C THEN LET LL=
    LL-C
320 LET C=C/2
330 PRINT AT 18,2+X*3;N;AT
    18,18+X*3;M
```

```
340 NEXT X
400 PRINT AT 21,6;"TOTAL
    HEX= ";CHR$ HI;CHR$ LO
500 RETURN
```

Cuando hayas tecleado todo el programa y lo ejecutes, observarás que se ponen a cero todos los números: binarios, decimales y hexadecimales. Si aprietas la tecla espaciadora aparecerá un 1 en cada base. Sigue apretando y el ordenador se pondrá a contar, sumando cada vez 1 a cada uno de los totales. Observa que el equivalente decimal se calcula sumando el valor de todo lugar que en binario tiene un 1.

El valor hexadecimal se calcula haciendo exactamente lo mismo, salvo que se toman cuatro dígitos a la vez.

Al pulsar la tecla B, se restará 1 de cada uno de los números y se ejecutarà el programa.

Puedes almacenar este programa y utilizarlo para convertir en hexadecimal números en binario siempre que te haga falta. La forma rápida de hacer esta conversión es pulsar cualquier teclado, con excepción de la barra de espacio o la B. En la pantalla aparecerá un signo de interrogación. Introduce cualquier número decimal que sea menor que 255, pulsa ENTER y te aparecerá su equivalente en binario y en hexadecimal.

Observarás que el número mayor que puede ser representado por un número de ocho bits en binario, es decir por un byte, es 11111111. Su equivalente decimal es 255, que en hexadecimal se escribe 255. Este es el mayor número posible de dos dígitos en hexadecimal. Cualquier número almacenado en un byte de la memoria de tu ordenador puede representarse en hexadecimal por medio de dos dígitos. Y todo el lenguaje máquina está construído enteramente a base de estos números hexadecimales de dos dígitos.

NUMEROS MAS GRANDES

Tu ordenador es capaz de manejar números mayores que 255, por el sencillo método de fraccionarlos en dos partes y almacenarlas en dos posiciones adyacentes de memoria. Esto te permitirá almacenar cualquier número hasta FFFF en hexadecimal, o lo que es igual, 65535 en decimal. FFFF es un número muy importante en los ordenadores domésticos, ya que es el máximo número de posiciones de memoria que se pueden direccionar.

Se pueden almacenar en memoria números aún mayores, fraccionándolos en tres o cuatro bytes y almacenándolos en direcciones de memoria sucesivas. En cuanto a la forma de almacenar dichos bytes, es una cuestión de convenio. En tu máquina se almacena el byte más bajo en la posición de memoria más baja y el byte alto en la posición más alta de memoria.

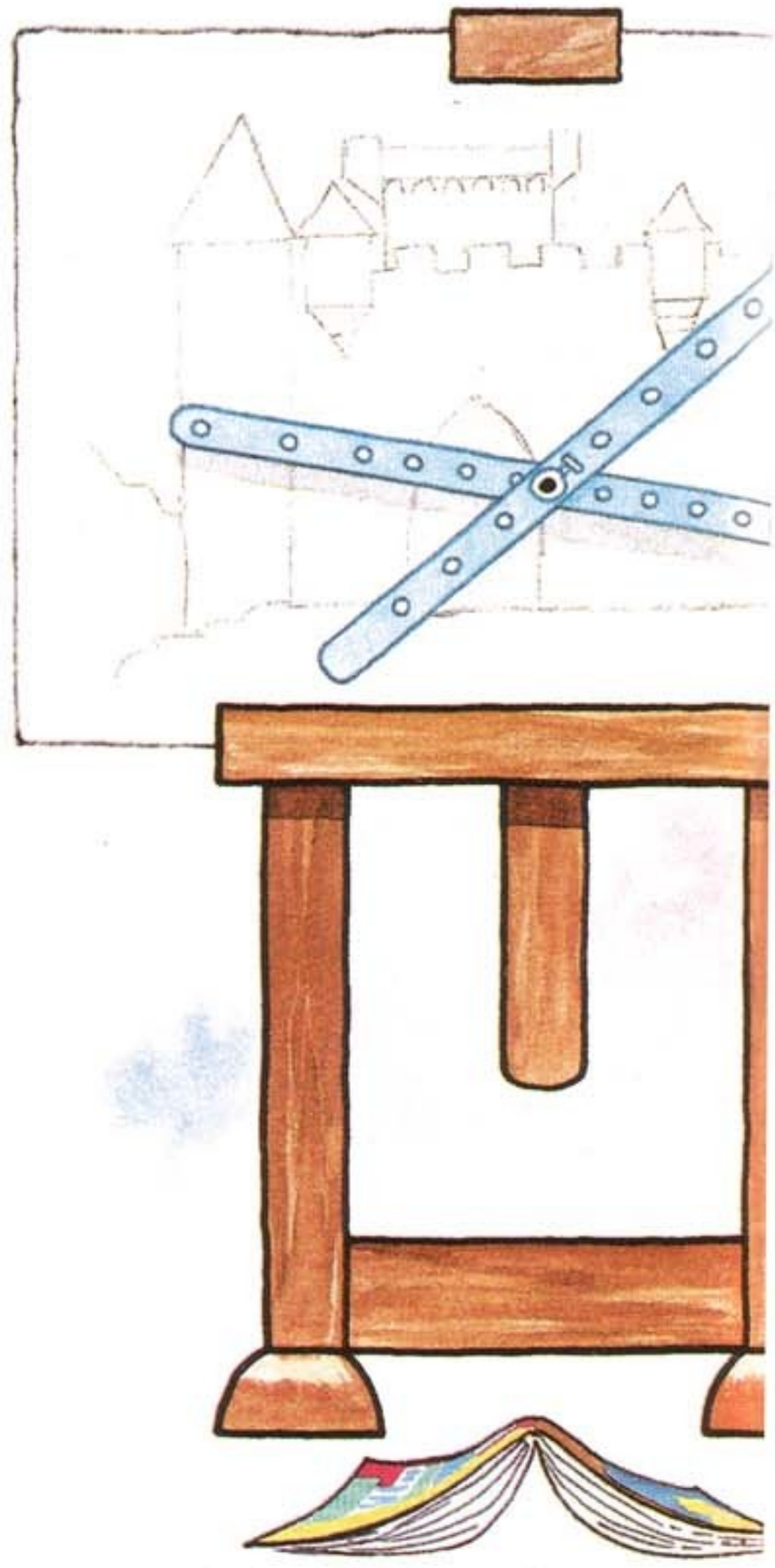
¿Pero de qué manera se representan en hexadecimal los números negativos? Nos ocuparemos de ello más adelante.

CONVERSION DE HEXADECIMAL A DECIMAL

Cada dígito sucesivo de un número expresado en forma hexadecimal tiene un peso 16 veces superior que el dígito que queda a su derecha. Así, para convertir en decimal un número como F6DA, se toma el número de la derecha y se convierte a decimal. Como es A, su valor es 10. El siguiente dígito de la izquierda tiene un peso 16 veces mayor, por lo que hay que pasarlo a decimal y multiplicarlo por 16. Al ser el número D, que vale 13, tenemos $13\times16=208$. El siguiente dígito por la izquierda tiene un peso 16 veces mayor, o sea, $6\times16\times16=1536$. El último dígito de la izquierda ha de multiplicarse aún por 16 una vez más. Como es el número F, que vale 15, tenemos: $15\times16\times16\times16=61440$. Así el número F6DA en hexadecimal será en decimal $10 + 208 + 1536 + 61440 = 63194$. También puedes utilizar el programa anterior para convertir los dos números hexadecimales al mismo tiempo y a continuación multiplicar el de la izquierda por 256.

Al ejecutar el programa se ponen a cero las tres líneas. Pulsa B y la línea de binario se llenará con 1. Más abajo la línea de decimal se llenará con potencias de dos. De derecha a izquierda obtienes 1, que es 2 elevado a la potencia cero, 2, que es 2 elevado a la primera potencia, 4, que es 2 al cuadrado, etc. La línea de hexadecimal funciona de la misma manera, excepto que los dos dígitos se calculan independientemente.

HARDCOPY DESDE CODIGO MAQUINA



Si posees una impresora tipo ZX-PRINTER **habrás observado el fácil manejo del comando COPY, pero también pronto te darás cuenta de las reducidas prestaciones que ofrece a cambio.**

El **Spectrum** viene preparado para funcionar directamente con su impresora propia, pero con las matriciales tenemos que valernos de un *interface* que gestione la comunicación. Así podemos obtener listados y copias de pantalla, pero siempre en el margen izquierdo.

Con este artículo pretendemos darte unas líneas generales para que, mediante unos ciertos conocimientos del código máquina y la previa adaptación a tu equipo, saques el mayor rendimiento a tu conjunto *interface*/impresora matricial. Nos centraremos en las posibilidades gráficas, presentando un programa en c.m. para realizar un *hardcopy* (copia de pantalla en papel) eligiendo la columna de impresión (una gran ventaja), y con la posibilidad de mezclarlo posteriormente con el texto —potenciando el uso de tu procesador de texto. No todo son listados...

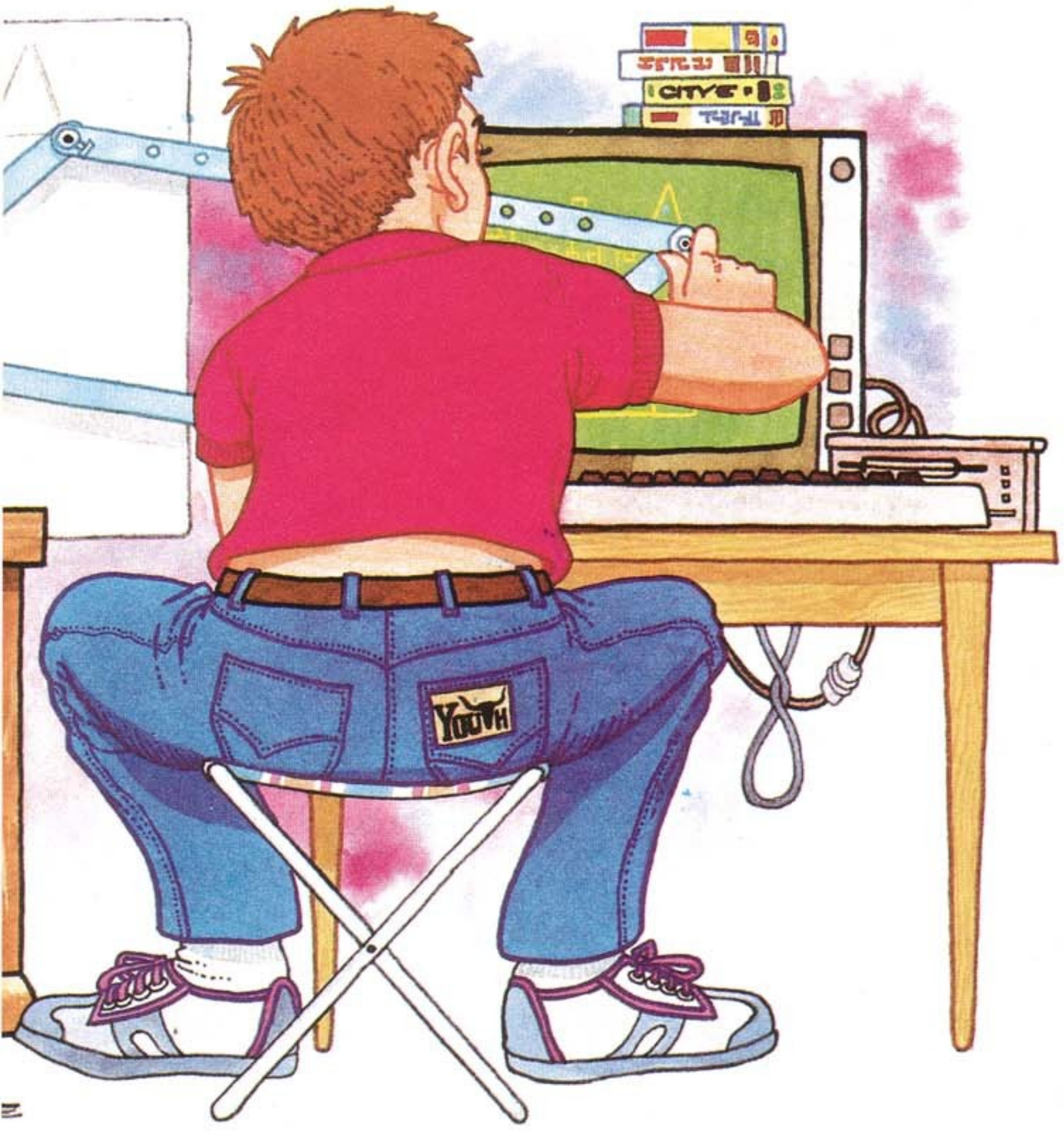
El primer problema que se nos presenta es el de la estructura de la memoria de pantalla, compuesta de 24 filas y 32 columnas. Las 24 filas quedan divididas en 3 tercios de 8 filas cada uno, que serán tratados de manera totalmente independiente en cuanto al manejo de los datos contenidos en cada uno de ellos. Las filas deben ser entendidas como hileras de caracteres, pero cada símbolo escrito ocupa 8 bytes en la memoria de pantalla, por lo que una fila de caracteres se compone a su vez de otras 8 pequeñas líneas de 32 bytes cada una. Así una fila de caracteres tendrá 32 * 8 = 256 bytes y cada tercio 256 * 8 = 2048 bytes es decir 2 Kytes cada uno. Así una pantalla entera tendrá 6 Kbytes, los 3 tercios (la memoria de atributos no se utiliza). También es de resaltar que una pantalla puede estar dibujada y no sólo escrita, por lo que cada caracater sería un «trocito» con 8 bytes de información del gráfico en pantalla.

Estamos acostumbrados a imprimir un símbolo en pantalla y que su aparición sea inmediata. La dificultad comienza cuando queremos recuperar eso datos insertos en la memoria, y utilizarlos para hacer, como en nuestro caso, un *hardcopy*. En cada uno de los citados tercios de pantalla —las primeras líneas de 32 bytes de cada una de las 8 filas de caracteres— son almacenadas una tras otra. Por lo tanto no se almacenan primero las 8 líneas de la primera fila, sino la primera línea de la primera fila, después al primera línea de la segunda fila, así hasta la octava, para seguir luego con la segunda línea de la primera fila, la segunda de la segunda fila, etc... Va saltando de 8 en 8 líneas hasta completar un tercio de pantalla, y después seguir rellenando los otros dos con el mismo sistema.

Todo lo expuesto se puede comprobar en el proceso de carga de una pantalla desde *cassette*. También se puede probar manejando la memoria de pantalla mediante POKE. Si esta parte de la memoria empieza en la posición 16384 ($4000), podemos probar CLS: POKE 16384,255 y conseguirás llenar el primer byte de la primera línea contenida en la primera fila. Para imprimir algo inmediatamente debajo lo lógico sería hacer POKE 16384+32,129, pero esto corresponde a la primera línea de la segunda fila. Lo correcto sería hacer: POKE 16384+(32*8),129 (hemos sumado a partir de la posición inicial los 32 bytes de las 8 primeras líneas de cada fila).

Debido a esta disposición de la memoria de pantalla hay otro factor importante a la hora de hacer el programa, y es que al copiar en papel las impresoras matriciales dividen cada caracter en 8 barras verticales de 8 *pixel* cada una, lo que ocasiona, sobre todo desde el c.m., tener que hacer una serie de bucles anidados un poco complejos. Para conseguir obtener cada una de las 8 barras verticales tenemos que tomar un *pixel* de cada uno de los 8 bytes definitorios del caracter representado (estos 8 bytes tienen unos disposición «horizontal», recuerda los POKEs anteriores). De esta manera como cada caracter o «trocito» de gráfico tiene sus bytes situados cada 8 líneas, para definir dicho gráfico necesitamos hacer 7 saltos/barra * 8 barras = 56 saltos.

De la idea anterior se desprende la necesidad de una forma rápida de realizar tan ingente número de saltos, y se impone el c.m.. Para contrastar



ideas comentamos el caso de la impresora del **Spectrum** que, en vez de ir tomando las barras verticales, se coloca al principio de cada línea de 32 bytes y va enviando la información a un *buffer* en el orden ya comentado. Cada vez que se llena el *buffer* con una línea, ésta es impresa.

La serie de programas que presentamos a continuación, están preparados para una impresora tipo **Seikosha**. Por si no dispones de una, intentaremos destacar los detalles diferenciadores más importantes para que puedas adaptar el programa fácilmente consultando su manual de instrucciones.

HARDCOPY EN CODIGO MAQUINA

El comentario de la rutina podría ser el siguiente:

• En primer lugar debemos abrir el canal hacia la impresora, el número 3 (líneas 90 y 100). Como recordarás si el canal abierto es el 2, todo uso de la instrucción RST $10 produce una impresión en pantalla.

• A partir de ahora comienzan una serie de bloques de datos que son enviados a la impresora y que son necesarios para el cometido que pretendemos. Toda esta información es enviada mediante la instrucción RST $10 que, como podremos comprobar, es una de las esenciales en esta rutina. Con esto y pocas instrucciones más nuestra rutina está casi terminada.

• El primer bloque al que hacemos referencia es el que elimina el espacio entre líneas (de 100 a 150). En nuestra copia de pantalla no necesitamos dicho espacio y por ello ejecutamos la instrucción de la impresora:

ESC L (27,76,N) donde N es el número que multiplicado por 1/18 nos da la magnitud en pulgadas del salto. En nuestro caso hacemos 27,76,2 por lo que el espacio entre líneas será 1/9 de pulgada, que será inapreciable. Una vez terminada la rutina debemos restablecer el valor inicial 27,76,3 por lo que la distancia entre líneas será mayor (1/6 de pulgada), espaciado normalmente utilizado en un proceso de escritura normal. Esta última acción es realizada por las líneas 680 a 730 del listado.

• Este envío de datos a la impresora desde c.m. será similar a la instrucción BASIC:

LPRINT CHR$ 27;CHR$76;CHR$ N (N=2 ó 3)

• A continuación viene la complicada estructura de bucles anidados que nos van a permitir ir tomando una a una todas las «barras» verticales constituyentes de cada caracter, con la salvedad de tener que ir tomando los bits de arriba hacia abajo, así el menos significativo será el tomado el primero (el más alto). En otros modelos de impresora, como las tipo **Epson** o **Admate** la mencionada barra vertical es tomada de abajo a arriba, siendo por tanto el menos significativo el mas bajo.

• En un principio cargamos en el registro doble DE la direccion de donde empezamos a tomar bytes, que lógicamente será el comienzo de la memoria de pantalla, la 16384 ($4000)

• El BUCLE0 abarca de la linea 170 a la 670. Es el que contiene a los demás y el encargado de que el proceso se repita tres veces, una por cada tercio de pantalla.

• El BUCLE1 (190-600) ejecuta el proceso 8 veces, una vez por cada una de las 8 filas del tercio.

• El BUCLE2 se repite 32 veces (el contador B es cargado con 32, LD, B,32) ya que cada fila posee 32 caracteres. Pero antes de empezar a repetir el proceso y cada vez que lo hagamos, hemos de enviar los datos a la impresora:

* La información para colocar el comienzo del *hardcopy* en el *pixel* «n» que varía dentro del rango comprendido entre 0 y 480 (el 480 es para una impresora matricial de 80 columnas,

pero los caracteres impresos son de 8*6 *pixels*, es decir 6 *pixels* de ancho, por lo que cuando hagamos una copia de pantalla escrita, al no utilizar el juego de caracteres de la impresora, nos dará la sensación de ser una letra más achatada aunque en realidad sea una copia de la letra del **Spectrum**). En este caso el valor elegido para n es 80, por tanto ejecutaremos la instrucción de la impresora :

ESC, POS,0,80 es decir, enviaremos los códigos 27,16,0,80. La ejecución de esta rutina es la que nos permite la comentada facilidad de elegir el lugar de impresión, pudiendo variarlo con sólo alterar los dos últimos números, que es la cifra de dos bytes conteniendo la dirección. Por lo tanto si queremos empezar a imprimir en el *pixel* 276 deberemos cambiar dichos datos por 27,16,1,20 ya que 276=1*256+20 (líneas 210 a 280 de nuestro listado). Una vez más insistimos en la necesidad de comprobación de los datos anteriores con el manual de la impresora en particular.

* A parte de lo anterior, la ejecución del BUCLE2 también requiere enviar a la impresora la información necesaria para especificar que los siguientes 256 caracteres a imprimir son bytes gráficos, es decir una fila de 32 caracteres con 8 bytes (o «barras») cada caracter. Esto se consigue ejecutando la instrucción de la impresora ESC G 1,0 o enviando por vía RST $10 los datos 27,71,1,0 (es decir 1*256+0). En nuestro listado las líneas 290 a 360.

• Para cada uno de los 32 caracteres debemos tomar los 8 bytes gráficos que lo constituyen. El BUCLE3 a parte de ser el encargado del proceso anterior a su vez contiene al BUCLE4 que se repertirá otras 8 veces para ir tomando cada bit de la memoria de la pantalla y que serán los constituyentes de las «barras». Estos dos últimos bucles son los que realmente se encargan de tomar los datos gráficos. Los anteriores eran la repetición un numero exacto de veces del proceso principal

• Los pasos principales a seguir son:

* Pasamos la direccion de referencia contenida en el registro DE al III

En un principio la dirección del DE será la 16384, pero luego irá cambiando (400,410).

* Giramos el contenido de la dirección apuntada por el registro HL (línea 440). Como ya sabemos el contenido de una dirección de memoria, es un byte que puesto en binario se compone de ceros y unos. Un uno se deberá traducir en un punto en el papel y el cero en un blanco. Tenemos que conseguir pasar esos ceros y unos a la impresora, pero el problema radica en la forma de hacerlo, ya que debemos tomar cada bit por separado de cada uno de los 8 bytes de un caracter, que a su vez están separados por 8 líneas. Para realizar el giro del contenido de la dirección elegida utilizamos la instrucción RLC (HL) (*Rotate Left Circular*) es decir realizamos un giro a la izquierda pasando el bit 0 a la posición del bit 1, el bit 1 a la del 2 y así hasta llegar al bit 7 que pasa a la posición del bit 0 y al carry. Por lo tanto si hacemos ahora un giro a la derecha en el registro A pero con la intervención del acarreo (RRA), conseguiremos pasar el contenido del bit 7 al A (el contenido del bit 7 fue pasado al acarreo). Para entenderlo mejor pondremos un ejemplo: Si el contenido de HL es 131 al hacer RLC (HL) quedará 131 → 10000011 → RLC → 00000111 acarreo → 1 si el registro A tiene 8 bits no especificados, al ejecutar RRA, conteniendo el acarreo un 1 quedará:

A → xxxxxxxx → RRA → 1xxxxxxx

No importa el contenido inicial de A, pues al repetirse el proceso 8 veces acabamos llenando dicho registro con 8 bits nuevos, que serán los bits de la «barra vertical».

* Una vez ejecutada la subrutina anterior incrementamos H (460), con lo que pasamos al byte 16384+256 en la primera línea de la segunda fila, lugar donde está situado el segundo byte constitutivo del caracter. Colocados aquí realizamos la misma operación de antes (RCL(HL), RRA) 8 veces (LD B,8 .... DJNZ BUCLE4) con lo que conseguimos llenar el acumulador con los valores deseados. A grandes rasgos este sería el sistema utilizado para el *hardcopy*. Sólo resaltar que en otras impresoras, como las **Epson** o **Admate**, el valor de los bits en la barra vertical se toma al revés, siendo el más significativo el más bajo debiendo por tanto cambiar las instrucciones 440 y 450 por:

440 BUCLE4 RRC (HL); Giro a la derecha

450 RLA ; Giro a la izquierda con carry

La línea 480 envía los datos obtenidos a la impresora. En las 510 y 520 (una vez definido el caracter) pasamos la dirección inicial guardada en el DE al HL. Incrementamos HL (530) con lo que pasaremos a la dirección de memoria contigua (no como al incrementar H que pasábamos 256 más adelante). En la 540 devolvemos el nuevo valor al DE con lo que habremos cambiado la dirección de referencia y estaremos listos para empezar a definir el caracter siguiente.

• Una vez finalizado el BUCLE2 (hemos enviado a la impresora una fila de caracteres) ejecutamos el código de la impresora LF (10), que indica que imprima todo lo hay en el *buffer*, vaya al final de línea y haga un retorno de carro (570,580). Hemos de aclarar que todo lo enviado a la impresora fue almacenado en el *buffer*, y no se imprime hasta que aparece la orden de «alimentación de línea» (*line feed*), LF (código 10).

• Una vez impreso un tercio de pantalla abrimos un pequeño bucle (BUCLE5) para pasar a la posición inicial del siguiente tercio. Para ello tomamos la posición de referencia del registro DE, la introducimos en HL (610 EXX DE,HL) e incrementamos H siete veces o 2 Kbytes menos 256 bytes. Más tarde se devuelve al DE (650).

NOTA: Para utilizar esta rutina con el **Interface I** o el **Discovery1** es necesario abrir el canal «b», que destokeniza los comandos del **Spectrum** (con estos *interfaces* utilizamos normalmente el canal «t»). Actúa de manera similar con otros *interfaces*.

```
10
20 ;  HARDCOPY GP250-X
30 ;        MOLISOFT
40
50
60          ORG  60000
70          ENT  60000
80          LD   A,3
90          CALL #1601
100         LD   A,27
110         RST  #10
120         LD   A,76
130         RST  #10
140         LD   A,2
150         RST  #10
160         LD   DE,#4000
170         LD   B,3
180 BUCLE0  PUSH BC
190         LD   B,8
200 BUCLE1  PUSH BC
210         LD   A,27
220         RST  #10
230         LD   A,16
240         RST  #10
250         LD   A,0
260         RST  #10
270         LD   A,80
280         RST  #10
290         LD   A,27
300         RST  #10
310         LD   A,71
320         RST  #10
330         LD   A,1
340         RST  #10
350         LD   A,0
360         RST  #10
370         LD   B,32
380 BUCLE2  PUSH BC
390         LD   B,8
400 BUCLE3  PUSH DE
410         POP  HL
420         PUSH BC
430         LD   B,8
440 BUCLE4  RLC  (HL)
450         RRA
460         INC  H
470         DJNZ BUCLE4
480         RST  #10
490         POP  BC
500         DJNZ BUCLE3
510         PUSH DE
520         POP  HL
530         INC  HL
540         EX   DE,HL
550         POP  BC
560         DJNZ BUCLE2
570         LD   A,10
580         RST  #10
```

```
590         POP  BC
600         DJNZ BUCLE1
610         EX   DE,HL
620         LD   B,7
630 BUCLE5  INC  H
640         DJNZ BUCLE5
650         EX   DE,HL
660         POP  BC
670         DJNZ BUCLE0
680         LD   A,27
690         RST  #10
700         LD   A,76
710         RST  #10
720         LD   A,3
730         RST  #10
740         RET
```

CARGADOR BASIC DE LA RUTINA HARDCOPY GP 250-X

La rutina es totalmente reubicable, y dada su corta longitud (100 bytes) puede pasar a formar parte de cualquier programa en c.m. De todas maneras adjuntamos el cargador BASIC, con la facilidad incluida de elección de la columna de impresión. La columna elegida debe estar entre la 0 y la 37. Si intentáramos imprimir en la 38, como la pantalla ocupa más de 42 columnas (256 bytes de largo), saldríamos del límite de las 80 columnas (480 bytes, porque cada columna tiene 6 *pixels* de largo). Además con este programa podemos imprimir esas dos últimas filas que siempre se nos resisten.

```
10 CLEAR #: OPEN #3;"B"
20 CLEAR 59999: FOR N=60000
   TO 60099: READ A: POKE N,
   A: NEXT N
30 INPUT  PAPER 6; INK 0;"
   COLUMNA DE IMPRESION?";A
40 IF A<0 OR A>37 THEN  GO
   TO 30
50 POKE 60033,INT (A*6)
60 LOAD ""SCREEN$ :
   RANDOMIZE USR 60000:
   PAUSE 0: GO TO 30
100 DATA 62,3,205,1,22,62,27
    ,215,62,76
200 DATA 215,62,2,215,17,0,
    64,6,3,197
300 DATA 6,8,197,62,27,215,
    62,16,215,62
400 DATA 0,215,62,80,215,62
    ,27,215,62,71
500 DATA 215,62,1,215,62,0,
    215,6,32,197
600 DATA 6,8,213,225,197,6,
    8,203,6,31
700 DATA 36,16,250,215,193,
    16,241,213,225,35
800 DATA 235,193,16,231,62,
    10,215,193,16,198
900 DATA 235,6,7,36,16,253,
    235,193,16,185
1000 DATA 62,27,215,62,76,
     215,62,3,215,201
```

INPUT sinclair

SERVICIO DE EJEMPLARES ATRASADOS

¡NO TE PIERDAS NI UN SOLO EJEMPLAR!

INPUT SINCLAIR quiere proporcionar a sus lectores este nuevo servicio de ejemplares atrasados para que no pierdan la oportunidad de tener en sus hogares todos los ejemplares de esta revista, líder en el mercado español.

Podréis solicitar cualquier número de **INPUT SINCLAIR** que querais, siempre al precio de cubierta (sin más gastos).

Utiliza el cupón adjunto, enviándolo a **EDISA** (Dpto. de Suscripciones), López de Hoyos, 141 - 28002 Madrid, o bien llámanos por teléfono al (91) 415 97 12.

INPUT sinclair

CUPON DE PEDIDO

SI, envíenme contrarreembolso ..... ejemplares de **INPUT SINCLAIR** de los números:

(marca con una (X) tu elección)

1	2	3	4	5	6	7	8	9	10	11

NOMBRE

APELLIDOS

DOMICILIO

NUM. PISO ESCALERA COD. POSTAL

POBLACION PROV.

TELEFONO FIRMA

TODO LO QUE SUBE, BAJA

- TRAYECTORIAS DE UN PROYECTIL
- DIFERENTES CAMPOS GRAVITATORIOS
- SIMULACION DE PARABOLAS
- CAMBIANDO EL ANGULO

«Lancé una flecha por los aires, cayó a tierra, no sé en qué punto». Aquí tienes la forma de hacer que tu ordenador haga por tí este cálculo, además de unas cuantas rutinas que pueden servirte para construir excitantes juegos de acción.

Hay una especial belleza en la contemplación de un ordenador mientras dibuja la trayectoria de un objeto volador, y existen muchos ejemplos —especialmente en la programación de juegos— en que resulta muy adecuado disponer de programas de este tipo. Si no se pudiera disponer de un ordenador, el dibujo de este tipo de trayectorias sería extraordinariamente tedioso y plagado de errores. Sin embargo, con sólo unas cuantas líneas de programa, puedes hacer que tu micro realice con precisión la gran cantidad de cálculos necesarios para contar con resultados sorprendentes.

La forma en que se mueven los objetos puede depender de varios factores, siendo importante que en un programa de acción se tengan en cuenta al menos algunos de ellos para conseguir un aceptable efecto de realismo.

Un caso importante muy especial es el de un objeto que abandona la superficie de la Tierra y está sometido a la ley de la gravedad. En este artículo veremos la manera de programar el movimiento de los proyectiles, entendiendo como tales los objetos que se mueven con velocidad horizontal constante (despreciando el rozamiento del aire) y verticalmente sometidos únicamente a la acción de la gravedad.

Una de las razones de por qué son tan comunes los juegos de combates que se desarrollan en el espacio, no sometidos a caída libre, es que en el espacio se pueden ignorar los efectos de la gravedad y del rozamiento del aire sobre el movimiento, tanto de los vehículos espaciales como de los proyectiles. Esto no quiere decir que en el espacio libre no exista fuerza gravitacional; claro que existe, lo que ocurre es que sus efectos son normalmente tan pequeños que se pueden ignorar dentro de un margen razonable de aproximación, o que se puede suponer que todos los objetos se ven afectados por un mismo campo gravitatorio.

En contraste con lo anterior, las batallas sobre la Tierra que implican el lanzamiento de proyectiles, han de tener en cuenta la gravedad y con frecuencia también la resistencia del aire y la acción del viento. Aparte de los juegos de tipo militar, hay otros muchos ejemplos en los que un conocimiento del movimiento de los proyectiles resulta esencial para una programación realista. Entre dichos ejemplos se encuentran muchos deportes y simulaciones de tipo atlético, incluyendo el lanzamiento de pelotas, buceadores, flechas y todo tipo de saltos.

Todos los proyectiles de estos ejemplos (ya se trate de personas o cosas) tienen una cosa en común: la trayectoria descrita responde a una familia de curvas conocidas como parábolas. No es difícil escribir un programa que simule el movimiento según una trayectoria parabólica; para ello no se requiere más que un entendimiento de las fuerzas que originan el movimiento y el uso de un poco de matemáticas elementales.

Por ejemplo, basta con saber que el movimiento de un balón al que se le da un puntapié, consiste en la combinación de dos movimientos en dos direcciones distintas, para poder abordar y resolver el problema de la programación de este movimiento. Una de estas direcciones es la del eje horizontal o eje X. La otra dirección es la del eje vertical o eje Y. A través de todo este artículo se supone que un proyectil se mueve a velocidad constante en la dirección del eje X. En realidad un objeto real se vería frenado por la acción de la resistencia del aire, pero esto es una complicación que es mejor evitar en general salvo en los trabajos de gran precisión.

MOVIMIENTO HORIZONTAL

Teclea el siguiente programa para simular el movimiento horizontal, pero no confundas «I» con «1».

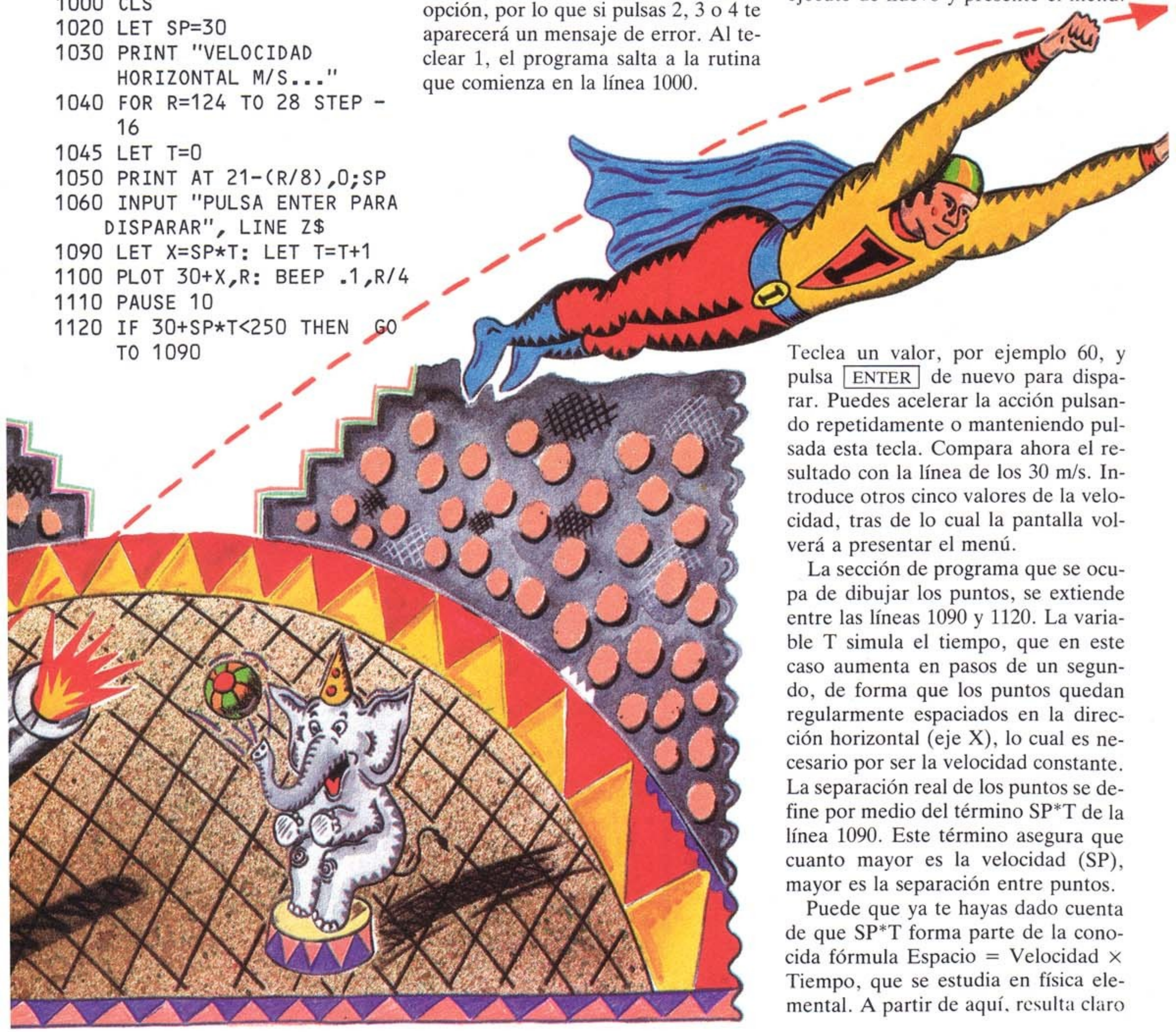
```
100 BORDER 7: PAPER 7: INK
  0: CLS
105 POKE 23658,8
110 PRINT  INVERSE 1;AT 2,12
    ;" MENU "
```

```
120 PRINT AT 6,5;"1:-
    MOVIMIENTO HORIZONTAL"
130 PRINT AT 8,5;"2:-
    MOVIMIENTO VERTICAL"
140 PRINT AT 10,5;"3:-
    MOVIMIENTO COMBINADO "
150 PRINT AT 12,5;"4:-
    ELEVACION"
200 LET I$=INKEY$: IF I$="" 
    THEN  GO TO 200
205 IF I$<"1" OR I$>"4" THEN
    GO TO 200
210 GO SUB VAL I$*1000
220 RUN
1000 CLS
1020 LET SP=30
1030 PRINT "VELOCIDAD
     HORIZONTAL M/S..."
1040 FOR R=124 TO 28 STEP -
     16
1045 LET T=0
1050 PRINT AT 21-(R/8),0;SP
1060 INPUT "PULSA ENTER PARA
    DISPARAR", LINE Z$
1090 LET X=SP*T: LET T=T+1
1100 PLOT 30+X,R: BEEP .1,R/4
1110 PAUSE 10
1120 IF 30+SP*T<250 THEN  GO
     TO 1090
1150 IF R=28 THEN GO TO 1200
1160 INPUT "NUEVA VELOCIDAD
     (O TERMINAR)", LINE I$
1165 LET SP=VAL I$
1170 IF SP<0 OR SP>1000 THEN
     GO TO 1160
1190 IF SP=0 THEN LET R=28
1200 NEXT R
1210 RETURN
```

Al ejecutar el programa te encuentras con un menú de cuatro opciones. De momento sólo tienes tecleada la rutina correspondiente a la primera opción, por lo que si pulsas 2, 3 o 4 te aparecerá un mensaje de error. Al teclear 1, el programa salta a la rutina que comienza en la línea 1000.

Dicha rutina utiliza un bucle FOR ... NEXT (líneas 1040 a 1200) que te permite lanzar un objeto horizontalmente a seis velocidades distintas. La primera vez que se recorre el bucle, se elige automáticamente una velocidad de 30 metros por segundo (m/s), la cual se simula como una serie de puntos a lo largo de una línea recta.

Después de que se ha hecho la primera pasada del bucle, el programa te invita a teclear un nuevo valor para la velocidad, pero puedes salir del bucle tecleando un 0 cuando el programa se ejecute de nuevo y presente el menú.



Teclea un valor, por ejemplo 60, y pulsa ENTER de nuevo para disparar. Puedes acelerar la acción pulsando repetidamente o manteniendo pulsada esta tecla. Compara ahora el resultado con la línea de los 30 m/s. Introduce otros cinco valores de la velocidad, tras de lo cual la pantalla volverá a presentar el menú.

La sección de programa que se ocupa de dibujar los puntos, se extiende entre las líneas 1090 y 1120. La variable T simula el tiempo, que en este caso aumenta en pasos de un segundo, de forma que los puntos quedan regularmente espaciados en la dirección horizontal (eje X), lo cual es necesario por ser la velocidad constante. La separación real de los puntos se define por medio del término SP*T de la línea 1090. Este término asegura que cuanto mayor es la velocidad (SP), mayor es la separación entre puntos.

Puede que ya te hayas dado cuenta de que SP*T forma parte de la conocida fórmula Espacio = Velocidad × Tiempo, que se estudia en física elemental. A partir de aquí, resulta claro

que el programa dibuja distancias (el espaciado entre puntos) para simular la velocidad.

MOVIMIENTO VERTICAL

Teclea ahora las siguientes líneas, que te proporcionarán una rutina para simular el movimiento en la dirección vertical:

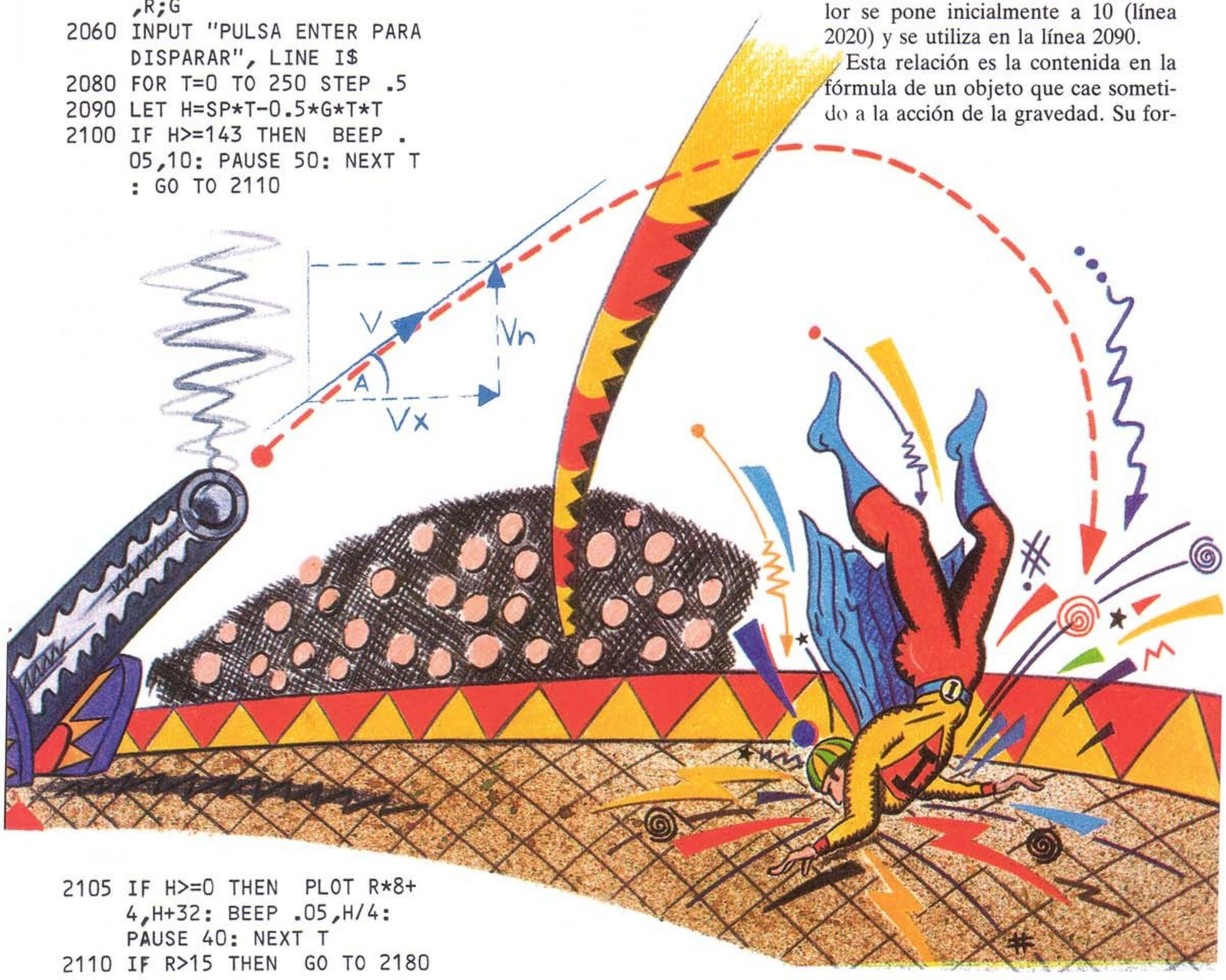
```
2000 CLS
2020 LET G=10: LET SP=50
2030 PRINT "ACELERACION GRA
     .M/S/S..."
2040 FOR R=3 TO 18 STEP 3
2050 PRINT AT 20,R;"*";AT 21
     ,R;G
2060 INPUT "PULSA ENTER PARA
     DISPARAR", LINE I$
2080 FOR T=0 TO 250 STEP .5
2090 LET H=SP*T-0.5*G*T*T
2100 IF H>=143 THEN  BEEP .
     05,10: PAUSE 50: NEXT T
     : GO TO 2110
2105 IF H>=0 THEN  PLOT R*8+
     4,H+32: BEEP .05,H/4:
     PAUSE 40: NEXT T
2110 IF R>15 THEN  GO TO 2180
2140 INPUT "NUEVA G (0 PARA
     TERMINAR)", LINE I$
2142 IF LEN I$=0 THEN  GO TO
     2140
2145 LET G=VAL I$
2150 IF NOT (LEN I$>0 AND G>
     =0 AND G<400) THEN  GO
     TO 2140
2170 IF G=0 THEN  LET R=18
2180 NEXT R
2190 RETURN
```

Haz correr el programa, tecleando ahora un 2 para seleccionar la segunda opción del menú. Pulsa ENTER y verás una serie de puntos dibujados verticalmente sobre la pantalla. Date cuenta no obstante de que los puntos no están regularmente espaciados, sino que se encuentran más agrupados cerca de la parte alta.

Esto se debe a que el movimiento del objeto se encuentra retardado por la acción de la gravedad. En esta ocasión el sonido te ayuda a explicar lo que está sucediendo. A medida que el objeto asciende, el tono del sonido se va elevando, mientras que cuando cae, su tono se va haciendo más bajo.

Igual que antes, la rutina te permite seis pruebas (línea 2040) con introducción de diferentes valores para apreciar el efecto de la gravedad. Esta es la misión de la variable G, cuyo valor se pone inicialmente a 10 (línea 2020) y se utiliza en la línea 2090.

Esta relación es la contenida en la fórmula de un objeto que cae sometido a la acción de la gravedad. Su for-



ma usual es $s = 1/2gt^2$, donde s es el espacio recorrido (en este caso H), t el tiempo y g el valor de la aceleración de la gravedad (G). Observa que en la línea 2090 se ha escrito T*T en lugar de T^2, ya que los ordenadores hacen con más rapidez la multiplicación que la elevación a potencias.

La aceleración de un objeto es una medida de lo que cambia su velocidad con el paso del tiempo. Cerca de la superficie de la Tierra, g tiene un valor aproximadamente igual a 10 m/s/s. Esto significa que la velocidad de un cuerpo que cae aumenta a razón de 10 m/s en cada segundo. Se trata de una aceleración negativa, ya que la línea 2090 contiene un signo menos (en lugar del signo más que aparece en la fórmula usual).

La expresión de g suele utilizarse habitualmente en relación con los viajes espaciales. Por ejemplo, la aceleración de un vehículo espacial tripulado que se lanza para entrar en órbita es próxima a los 10 g. Esto significa que dicho vehículo espacial aumenta su velocidad a razón de 10*10 m/s/s, es decir·100 m/s^2. En un posterior artículo nos ocuparemos con mayor detalle del tema de las órbitas.

La primera vez que se recorre el bucle que empieza en la línea 2040, se dibujan las posiciones del objeto a intervalos de un segundo. La velocidad inicial es de 50 metros por segundo y el valor de G es de 10 m/s^2; ambos valores se establecen en la línea 2020. Igual que en la rutina anterior, puedes acelerar la acción manteniendo pulsada o pulsando repetidamente ENTER. Puedes cambiar entonces el valor de G, cuando te lo indique el mensaje de pantalla, y comparar el efecto de seis valores diferentes. Antes de que se complete el bucle, puedes salirte de él introduciendo un 0 cuando el programa retorna al menú.

MOVIMIENTOS COMBINADOS

Para simular el movimiento de un proyectil, sólo tienes que combinar estas rutinas de forma que el objeto se mueva al mismo tiempo en las dos direcciones, horizontal y vertical. Teclea las siguientes líneas, y tendrás la tercera rutina:

```
3000 CLS
3020 LET G=10: LET SP=50
3030 FOR R=1 TO 6
3040 PRINT AT 0,0;"G=";G;"M
     /S/S"'"VELOCIDAD ";SP;
     "M/S "
3050 INPUT "PULSA ENTER PARA
     DISPARAR", LINE I$
3070 FOR T=0 TO 250 STEP .5
3080 LET H=SP*SIN ((PI/180)
     *45)*T-.5*G*T*T
3090 LET X=SP*COS ((PI/180)*
     45)*T
3100 IF H<175 AND X<255 THEN
     GO TO 3105
3102 IF H>0 THEN  BEEP .05,
     10: PAUSE 25: NEXT T:
     GO TO 3110
3103 LET T=250: NEXT T: GO
     TO 3110
3105 IF H>=0 THEN  PLOT X,H
     : BEEP .05,H/4: PAUSE
     40: NEXT T: GO TO 3110
3106 LET T=250: NEXT T
3110 IF R=6 THEN  GO TO 3230
3120 PAUSE 50
3140 INPUT "NUEVA G(O PARA
     TERMINAR)", LINE I$
3150 IF LEN I$=0 THEN  GO TO
     3140
3155 LET G=VAL I$
3160 IF NOT (LEN I$>0 AND G>
     =0 AND G<1000) THEN  GO
     TO 3140
3170 IF G=0 THEN  LET R=6:
     GO TO 3230
3190 INPUT "NUEVA VELOCIDAD"
     , LINE I$
3195 LET SP=VAL I$
3200 IF NOT (LEN I$>0 AND SP
     >0 AND SP<1000) THEN
     GO TO 3190
3230 NEXT R
3240 RETURN
```

Ejecuta el programa, tecleando un 3 como respuesta al mensaje de pantalla, seleccionando así la tercera opción. Cuando pulses ENTER, van apareciendo puntos en una curva que arranca en la parte inferior izquierda de la pantalla y termina en algún punto de la parte inferior derecha. Esta es la trayectoria de un objeto disparado a una velocidad de 50 m/s con un valor de la gravedad de 10 m/s^2.

La estructura de la rutina es análoga a las de las dos anteriores. Las secciones encargadas del cálculo y del dibujo están en un bucle FOR ... NEXT (líneas 3030 a 3230), que te permite hacer una comparación de seis trayectorias posibles, cinco de las cuales son especificadas por tí. Al igual que en las pruebas anteriores, puedes salirte de la rutina tecleando un 0 para volver al menú. Sin embargo es más probable que quieras introducir un nuevo juego de valores para comparar las trayectorias.

Teclea un valor de 5 para G, manteniendo el valor de SP en 50, y a continuación pulsa ENTER para disparar. Esta vez el objeto se moverá con mayor rapidez y llegará más lejos. Mantén ahora el valor de G a 5, pero reduciendo SP a 25 y compara los resultados. Continúa la experimentación cambiando G y SP, escuchando los sonidos, para ayudarte a entender el movimiento del objeto cuando desaparece de la pantalla. Una nota cuya altura aumenta indica que el objeto está subiendo, mientras que cuando el tono baja, indica que el objeto está cayendo.

FUNCIONAMIENTO

Recuerda que la trayectoria del objeto se dibuja con coordenadas H en la dirección del eje Y, y coordenadas X en la dirección del eje X. Estas dos coordenadas se calculan en las líneas 3080 y 3090. La única diferencia es que aquí la coordenada H contiene la velocidad (SP) multiplicada por el seno de un ángulo, y la coordenada X contiene la SP multiplicada por el coseno de ese mismo ángulo (45°). Esto explica porqué cuando G es pequeño y SP es grande, la trayectoria aparente es una línea diagonal a 45°.

La razón por la que se emplean estas funciones trigonométricas es el cálculo de la fracción del movimiento del objeto que corresponde a cada una de las dos direcciones: vertical y horizon-

tal. A estas fracciones se les llama las componentes. Si por ejemplo la velocidad inicial del objeto es 50 m/s, las componentes horizontal y vertical son menores que 50, si bien su suma da exactamente 50 m/s.

No te equivocarás si recuerdas que la componente vertical es SP*SIN A y la componente horizontal es SP*COS A. La variable A es el ángulo de elevación del cañón, el arco o lo que sea.

Para entender cómo se llega a la obtención de estos valores, te será de gran ayuda una figura. La segunda ilustración de este artículo muestra un proyectil que inicia su movimiento con una velocidad real V y un ángulo A con la horizontal. Las líneas de puntos muestran los componentes de la velocidad (Vh y Vx) en las dos direcciones. El seno del ángulo A es Vh/V, relación que puede escribirse de forma equivalente como Vh = V*SIN A. Análogamente, el coseno de A es Vx/V, lo que es equivalente a escribir Vx = V*COS A.

Siguiendo con esta misma idea en nuestro programa, necesitas calcular SP*SIN 45 para la componente vertical de la velocidad y SP*COS 45 para la componente horizontal.

CAMBIO DE ANGULO

Llegados a este punto, puede que te extrañe el por qué ha de fijarse el valor del ángulo en 45°, ya que ello limita el alcance del proyectil. Tanto el ángulo de elevación como la velocidad inicial pueden modificarse para cambiar el alcance de los proyectiles. Y es más corriente modificar únicamente el ángulo para cambiar la distancia, manteniendo constante la velocidad inicial. Esto es precisamente lo que hace la siguiente rutina:

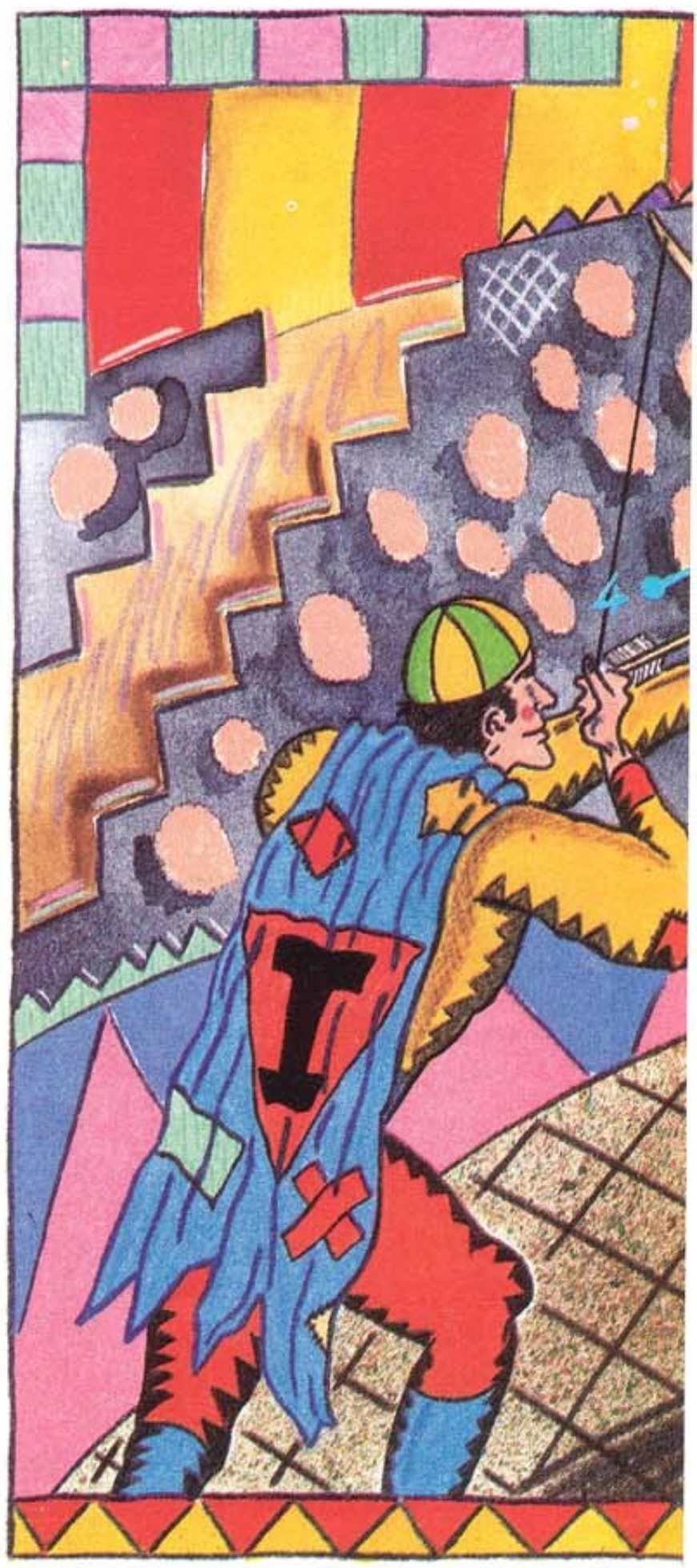
```
4000 CLS
4010 LET FL=0
4020 RESTORE : FOR N=0 TO 7:
     READ A: POKE USR "A"+N,
     A: NEXT N
4040 LET A=70: LET SP=50
4060 PRINT AT 0,0;"ANGULO=";
     A;CHR$ 144;CHR$ 32
4070 INPUT "PULSA ENTER PARA
     DISPARAR", LINE I$
4080 FOR T=0 TO 250 STEP .5
4090 LET H=SP*SIN ((PI/180)*
     A)*T-.5*10*T*T: LET X=
     50*COS ((PI/180)*A)*T:
4100 IF H>=0 THEN  PLOT X,H
     +16: BEEP .05,H/4:
     PAUSE 40: NEXT T: GO TO
     4110
4105 LET T=250: NEXT T
4110 PAUSE 50
4130 INPUT "NUEVO ANGULO(0
     PARA TERMINAR)", LINE I$
4135 IF LEN I$=0 THEN  GO TO
     4130
4140 LET A=VAL I$
4150 IF NOT (LEN I$>0 AND A>
     =0 AND A<90) THEN GO TO
     4130
4160 IF A=0 THEN  LET FL=1
4170 IF NOT FL THEN  GO TO
     4060
4180 RETURN
5000 DATA 24,36,36,36,24,0,
     0,0
```

Al ejecutar la cuarta opción, verás la trayectoria seguida por un objeto lanzado con una velocidad de 50 m/s y con un ángulo de elevación de 70° (ambos valores se establecen en la línea 4040 y se utilizan en las relaciones de la línea 4090).

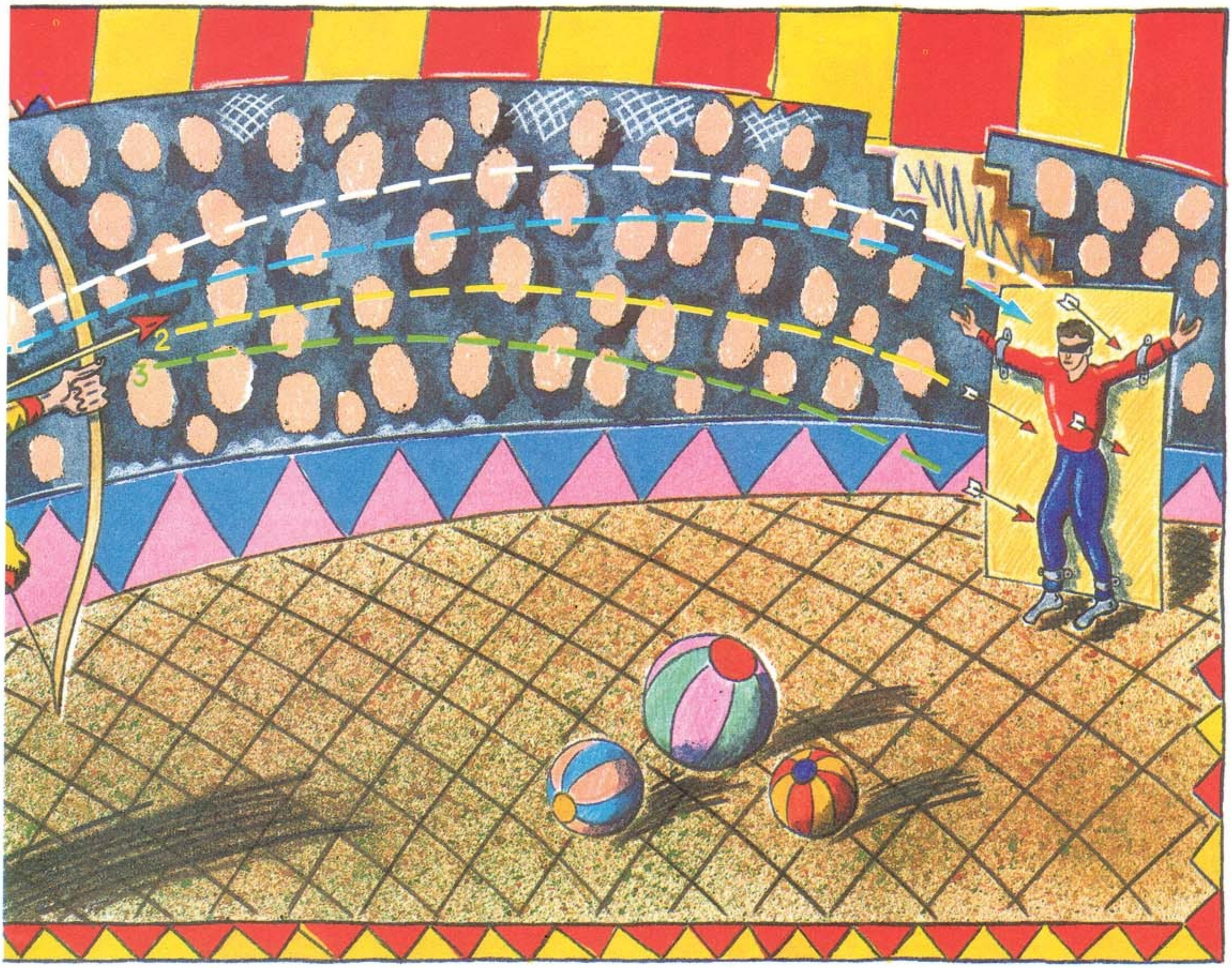
Esta rutina funciona igual que la anterior, si bien aquí puedes introducir los valores de los ángulos que quieras sin tener que volver al menú. El ángulo ha de estar comprendido entre 1° y 89°. Así, cada vez que ejecutes la rutina, se le asigna un valor a la variable A de la línea 4090, correspondiente al ángulo que hayas tecleado. Esta vez la rutina contiene un bucle infinito, por lo que la pantalla no retornará al menú a menos que introduzcas 0 como valor del ángulo.

Utilizando esta rutina, trata de encontrar el ángulo con el que se obtiene el mayor alcance, es decir el que permite que el objeto llegue más lejos en dirección horizontal. Fácilmente descubrirás que este ángulo es de 45 grados.

Casi todo el mundo conoce este resultado de forma intuitiva o a partir de



su propia experiencia, pero lo que ya es menos conocido es que en la práctica la resistencia del aire hace que haya una diferencia significativa en el resultado en muchos casos, y que el valor de 45 grados sólo permite obtener el alcance máximo cuando los puntos de partida y de caída están a la misma altura. Sin embargo, cualquier juego de proyectiles que dependa sólo de que el jugador obtenga el mayor alcance, está condenado al fracaso, ya que no se producirá una variación sustancial respecto al caso ideal que la mayoría de los jugadores conocen o sospechan. En un artículo posterior veremos la forma en que pueden utilizarse estas rutinas como base para un desafiante y cautivador juego.



GANADORES DE LOS MEJORES DE INPUT SINCLAIR

En el sorteo correspondiente al número 11 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
J. Ignacio Gude Basterrechea	Madrid	Movie
Miguel Gómez Bataller	Valencia	Vay of the tiger
Jesús Pelegrín Plazas	Lorca (Murcia)	Phantomas
Jorge Dieznandino	Algorta (Vizcaya)	Profanation
Jorge Gutiérrez Repiso	Málaga	Kung fu master
Fernando Fanes Trillo	Barcelona	Green Beret
Angel Fabián Brea Rodríguez	Chiclana (Cádiz)	Ping Pong
Joan Grau Burgués	Lérida	Winter Games
Alberto Redondo González	Madrid	Back to the future
Martín Offmann	Tarragona	Phantomas

Programación

PONIENDO LAS COSAS EN ORDEN

- ¿QUE ES LA ORDENACION?
- EL METODO DE LA BURBUJA
- LOS MODULOS DE SHELL Y SHELL-METZNER
- MEJORA DE LA VELOCIDAD

Todo programa que tenga que manejar datos, puede beneficiarse del uso de una rutina de ordenación del tipo que sea. Aquí nos ocuparemos de tres de los métodos de ordenación más comunes.

La ordenación es uno de los aspectos más básicos del proceso de la información. Los ordenadores son muy buenos para la manipulación de datos con gran rapidez y la ordenación juega un papel muy importante al hacer que esta información sea fácilmente accesible para su inspección y corrección.

Imagínate lo difícil que sería seguirle la pista a cualquier voz en particular dentro de un libro si éste no tuviera índice. O consultar un número de teléfono en una guía que no estuviera ordenada alfabéticamente. Ambas son listas esenciales de información que, a diferencia de las entradas aleatorias de algo en una lista de la compra, contienen sus elementos ordenados siguiendo un determinado criterio, en este caso el alfabético. Pero también podrías fijar tu atención en una lista de elementos ordenados numéricamente, como por ejemplo las entradas de cheques en una sucursal bancaria.

Hay muchos tipos de programas en los que se utilizan rutinas de ordenación; también ocurre esto en los juegos en los que se hace la clasificación con arreglo a una puntuación.

Pero es más corriente encontrar rutinas de ordenación en los programas han de manejar la información en listas, tales como listas de direcciones, datos personales, etc.

¿QUE ES LA ORDENACION?

Se llama ordenación o clasificación simplemente a colocar la información en un orden especificado, tal como se hace cuando se abre en abanico un grupo de cartas y se van poniendo por orden de números y palos.

En las ordenaciones de tipo numérico, aparecen sucesivamente valores mayores antes o después que valores más pequeños. En las de tipo alfabético, aparecen letras más próximas al comienzo del alfabeto antes o después que otras letras más lejanas. También puedes basar una ordenación en un criterio alfanumérico, en el que se consideran letras y números, aunque siempre debe prevalecer uno de ambos elementos sobre el otro; normalmente las letras tienen más prioridad que los números.

En las ordenaciones numéricas, el valor 1 es el que tiene usualmente la máxima prioridad, mientras que en las alfabéticas es la A el elemento de máxima prioridad. Pero para determinados fines, estos criterios pueden cambiarse, asignando por ejemplo al valor 9 y a la letra Z la máxima prioridad.

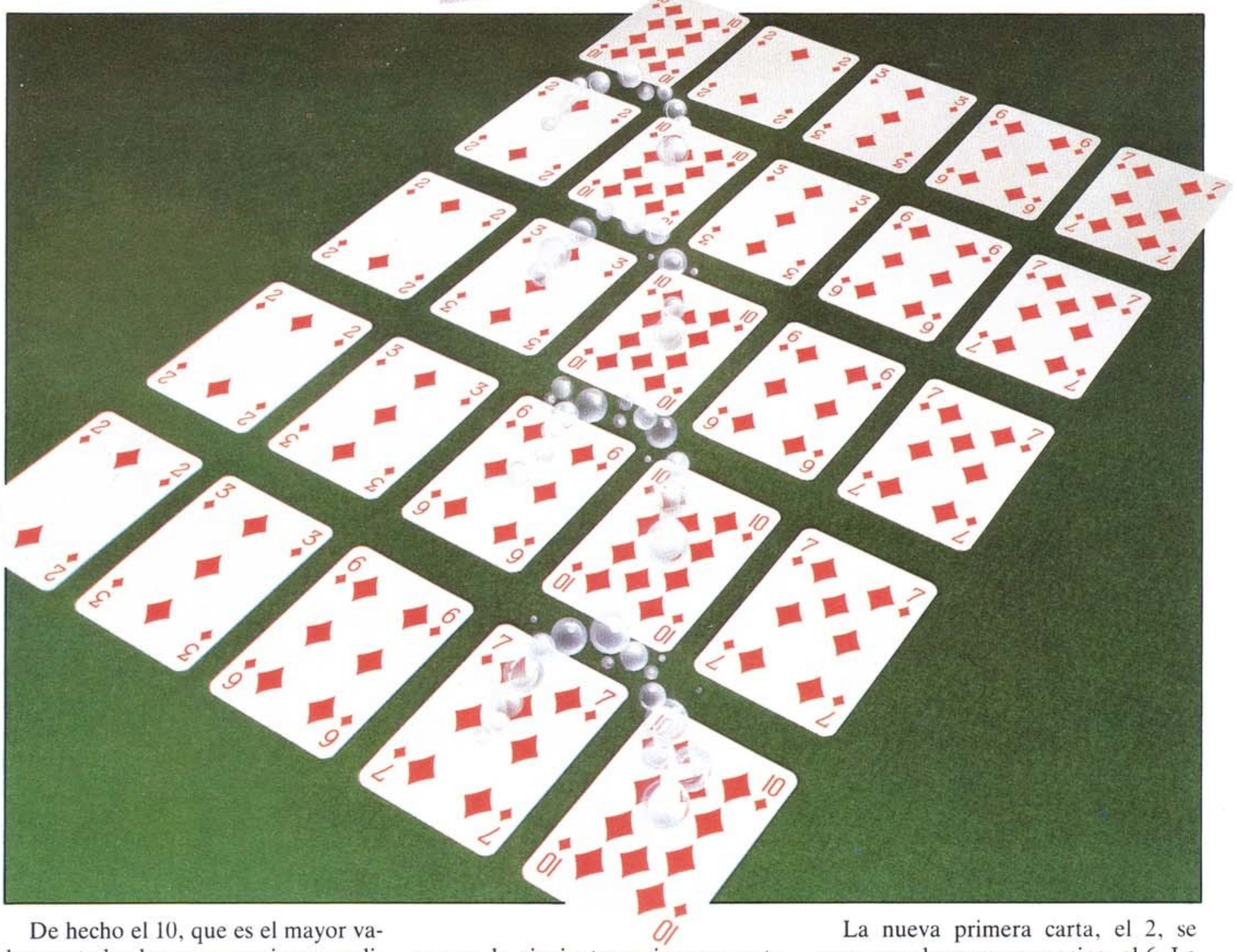
ORDENACION POR EL METODO DE LA BURBUJA

La forma más sencilla de rutina de ordenación conocida se suele llamar ordenación por el método de la burbuja. Examinemos ahora el proceso con detalle. Utiliza como ejercicio unos cuantos naipes, o corta y numera unos cuantos trozos de papel.

Coloca esta selección de cartas en una mesa por este orden, con el «tres» en la posición más alejada de tí:

```
ULTIMO          [TRES]
                [SIETE]
                [DOS]
                [SEIS]
PRIMERO         [DIEZ]
```

En el método de ordenación de la burbuja, el primer valor —en este caso la carta número 10— se compara con la que figura a continuación de ella en la lista, permutándolas cuando la siguiente es de menor valor (como ocurre en este caso). El 10 se compara entonces con el 2, con el 7 y con el 3. Todas las veces se permuta el 10 con las que se va encontrando, debido a que siempre es el mayor de los dos valores que se comparan.



De hecho el 10, que es el mayor valor en todas las comparaciones realizadas, ha «burbujeado» a través de los restantes valores, lo que explica el nombre por el que se conoce popularmente a esta rutina de ordenación. Por la misma razón, los valores más pequeños burbujean hacia abajo. Esta ordenación de la burbuja se suele llamar ordenación por cambio u ordenación por intercambio.

Así, después de una pasada, la fila de cartas quedaría de la siguiente forma:

```
ULTIMO      [DIEZ]
            [TRES]
            [SIETE]
            [DOS]
PRIMERO     [SEIS]
```

El proceso comienza de nuevo ahora con la siguiente «primera» carta, que es el 6, el cual se compara con el 2 y se permuta debido a que es mayor. Pero la siguiente comparación se hace con el 7. Ahora, como 7 es mayor, la ordenación continúa con el 7, dejando el 6 en la última posición en la que obtuvo una comparación con ventaja. La carta número 7 se compara primero con el 3 y después con el 10, desplazándose hacia arriba únicamente un lugar, debido a que se encuentra con un valor mayor que el suyo. El nuevo orden de las cartas es ahora el siguiente:

```
ULTIMO      [DIEZ]
            [SIETE]
            [TRES]
            [SEIS]
PRIMERO     [DOS]
```

La nueva primera carta, el 2, se compara ahora con su vecina, el 6. La carta número 6 es la que prevalece ahora por ser mayor, permutándose con la número 3, pero ya no puede llegar más lejos al compararse con la carta número 7. La ordenación llega entonces a su fin al no poder producirse más intercambios, con lo que el orden final de las cartas queda de la siguiente manera:

```
ULTIMO      [DIEZ]
            [SIETE]
            [SEIS]
            [TRES]
PRIMERO     [DOS]
```

Un ordenador realizaría de hecho una última pasada por esta última serie ordenada, para ver si hay algún par más que admita un intercambio, dete-

niéndose únicamente al no poder hacer ninguno más.

En el curso de una ordenación por burbuja, se produce un determinado número de comparaciones e intercambios. Estas dos operaciones diferentes forman parte de toda rutina de ordenación, y la diferencia entre unas rutinas y otras depende enteramente del número de estas operaciones y del tiempo requerido.

Veamos un ejemplo utilizando la siguiente sucesión de números:

```
ARRIBA:                ABAJO:
67 35 72 19 47 38 11 96
```

El primero de esta lista (arriba) es el 67. Este valor se compara con 35 y se intercambia con él, pero ya no se mueve más. El valor mayor que interviene en las comparaciones es ahora 72, se compara y se va permutando con 19, 47, 38 y 11 hasta que se encuentra con 96, momento en que la rutina vuelve al principio comenzando con su nuevo primer número. En la tabla 1 se muestra el proceso paso a paso, se presentan encerrados entre paréntesis los elementos que van siendo sometidos a comparación. Si en la fila siguiente aparece un número diferente en esa misma columna, es que se ha producido un intercambio.

¿Cómo sería entonces un programa de ordenación por el método de la burbuja? Ensaya el siguiente programa en el que la rutina de ordenación de la burbuja ocupa la subrutina que empieza en la línea 1000. Almacena el programa, al que podrás añadir después otras rutinas de ordenación.

Teclea

```
10 POKE 23658,8: LET T=0:
   INPUT "NUMERO DE CONCEPTOS
   ";AA: IF AA<2 THEN  GO TO
   10
15 DIM A(AA)
20 PRINT "TABLA DESORDENADA"
   : PRINT
30 FOR Z=1 TO AA
40 LET A(Z)=INT (RND*100)+1
50 PRINT TAB T;A(Z);: LET T=
   T+4: IF T>30 THEN  LET
   T=0
60 NEXT Z
70 PRINT : PRINT : PRINT
   "PULSA 0 PARA ORDENAR"
80 LET K$=INKEY$: IF K$<>"0"
   THEN  GO TO 80
90 GO SUB 1000
100 PRINT : PRINT "TABLA
    ORDENADA": PRINT
110 LET T=0: FOR Z=1 TO AA
120 PRINT TAB T;A(Z);: LET T
    =T+4: IF T>30 THEN  LET
    T=0
130 NEXT Z
140 GO TO 10
999 REM METODO DE BURBUJA
1000 FOR Z=1 TO AA-1
1010 LET ZZ=0
1020 FOR Y=1 TO AA-Z
1030 IF A(Y+1)<A(Y) THEN LET
     X=A(Y): LET A(Y)=A(Y+1)
     : LET A(Y+1)=X: LET ZZ=1
1040 NEXT Y
1050 IF ZZ=0 THEN  RETURN
1060 NEXT Z: RETURN
```

Teclea RUN para ejecutar el programa. Empieza preguntándote cuántos elementos quieres ordenar, creando un conjunto de números pseudoaleatorios basado en tu respuesta. Estos números se presentan en la pantalla bajo el encabezamiento «Tabla sin ordenar». A continuación aparece un mensaje preguntándote si estás dispuesto para empezar la ordenación. Para empezar no tienes más que teclear «C».

Se produce entonces la ordenación. Con unos pocos elementos puede que no tarde más que un segundo o poco más, pero prueba a contestar al mensaje de apertura con el número 50, por ejemplo. Una vez que se haya com-

TABLA 1

Arriba							Abajo	Arriba							Abajo
(67)	(35)	72	19	47	38	11	96	19	35	38	11	47	(67)	(72)	96
35	(67)	(72)	19	47	38	11	96	19	35	38	11	47	67	(72)	(96)
35	67	(72)	(19)	47	38	11	96	(19)	(35)	38	11	47	67	72	96
35	67	19	(72)	(47)	38	11	96	19	(35)	(38)	11	47	67	72	96
35	67	19	47	(72)	(38)	11	96	19	35	(38)	(11)	47	67	72	96
35	67	19	47	38	(72)	(11)	96	19	35	11	(38)	(47)	67	72	96
35	67	19	47	38	11	(72)	(96)	19	35	11	38	(47)	(67)	72	96
(35)	(67)	19	47	38	11	72	96	19	35	11	38	47	(67)	(72)	96
35	(67)	(19)	47	38	11	72	96	19	35	11	38	47	67	(72)	(96)
35	19	(67)	(47)	38	11	72	96	(19)	(35)	11	38	47	67	72	96
35	19	47	(67)	(38)	11	72	96	19	(35)	(11)	38	47	67	72	96
35	19	47	38	(67)	(11)	72	96	19	11	(35)	(38)	47	67	72	96
35	19	47	38	11	(67)	(72)	96	19	11	35	(38)	(47)	67	72	96
35	19	47	38	11	67	(72)	(96)	19	11	35	38	(47)	(67)	72	96
(35)	(19)	47	38	11	67	72	96	19	11	35	38	47	(67)	(72)	96
19	(35)	(47)	38	11	67	72	96	19	11	35	38	47	67	(72)	(96)
19	35	(47)	(38)	11	67	72	96	(19)	(11)	35	38	47	67	72	96
19	35	38	(47)	(11)	67	72	96	11	19	35	38	47	67	72	96
19	35	38	11	(47)	(67)	72	96								

pletado el proceso de ordenación, se presentará el nuevo grupo de números junto a los otros.

Para poder comparar unos métodos con otros, es útil tomar el tiempo requerido por el proceso de ordenación. El cronometraje es algo que puede hacer muy bien el ordenador por tí. Incorpora esta rutina de temporización dentro del programa existente:

Teclea

```
90 POKE 23672,0: POKE 23673
   ,0: GO SUB 1000: PRINT :
   PRINT (PEEK 23672+256*
   PEEK 23673)/50;"
   SEGUNDOS"
```

Para que empiece el proceso de medida de tiempos, y la rutina de ordenación, pulsa la tecla C cuando aparezca el mensaje de comienzo. El tiempo transcurrido hasta el momento en que se completa la rutina aparece en la pantalla y la ejecución del programa arranca de nuevo automáticamente.

Aunque los tiempos de ordenación son probablemente mucho más rápidos que los que podrías lograr ordenando tú directamente los números, para las magnitudes de tiempo que se manejan con los ordenadores se trata de un tiempo desesperadamente largo. Por esta razón no se utiliza normalmente dicha rutina en este estado tan «crudo» en los programas de proceso de datos, a no ser que se trate de listas muy cortas.

Pero asombrosamente existe un ejemplo en el que la rutina del método de la burbuja proporciona realmente los resultados más rápidos. Se trata del caso en que se utiliza para reordenar una lista ya ordenada en la que se ha agregado un nuevo elemento.

Por ejemplo en un programa de direcciones postales, las nuevas direcciones se ordenan separadamente y después se mezclan con la lista vieja. Pero también se pueden ir añadiendo direcciones de una en una a la lista vieja, sometiendo después todo el lote a una nueva ordenación.

Bajo estas circunstancias, el método de la burbuja no tiene que hacer nada más que comparar la nueva dirección con todos sus vecinos hasta que llegue a su posición correcta. Y esto puede hacerse muy rápidamente, por lo que se utiliza muchas veces con buen resultado dentro de rutinas de ordenación de otro tipo.

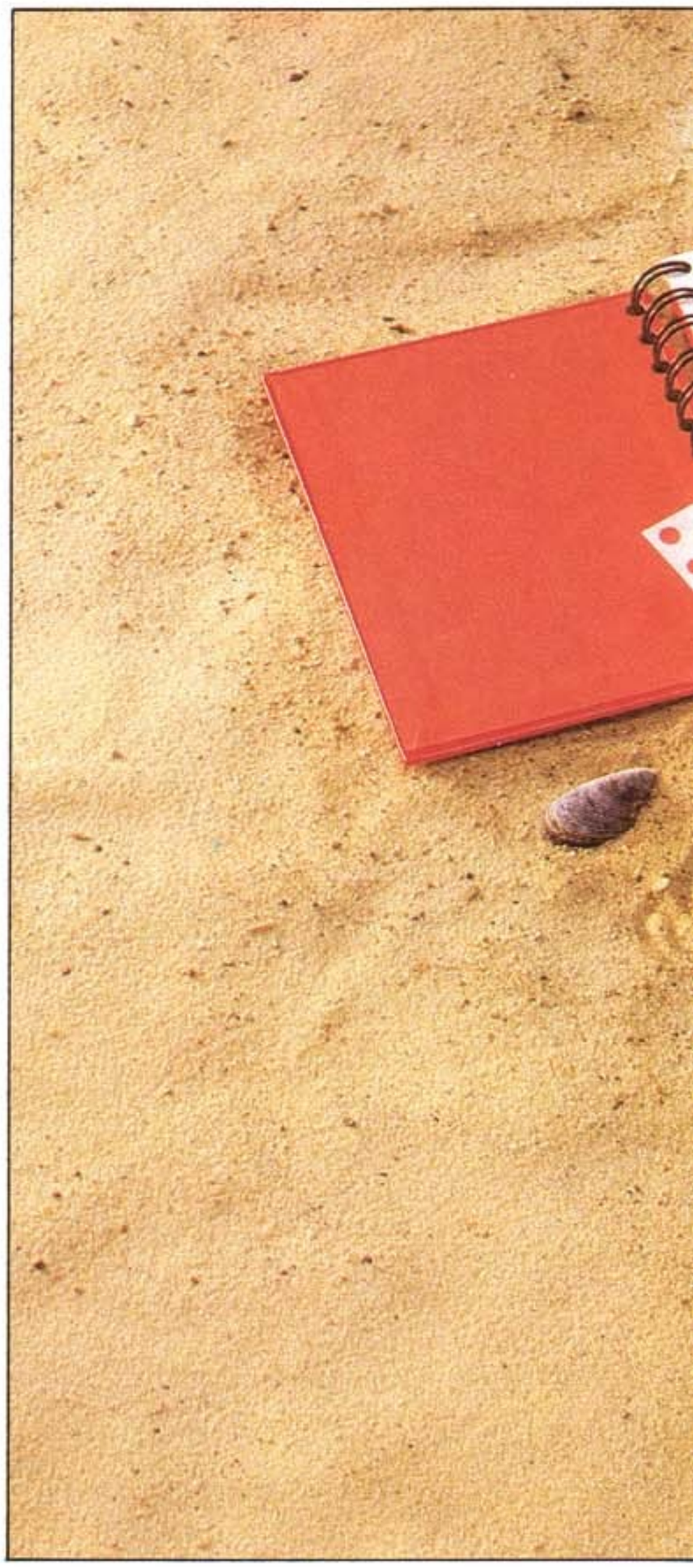
La velocidad de todas las rutinas de ordenación puede aumentarse considerablemente si la información se presenta ya de alguna forma parcialmente ordenada. Esto es especialmente cierto en el caso sencillo de la ordenación por el método de la burbuja.

Por ejemplo, si los elementos tienen que ser ordenados por orden de fechas, será muy útil introducir los datos en la forma Año/Mes/Día en vez de la forma más usual Día/Mes/Año. La rutina podría ocuparse de los años con mucha rapidez, para generar una primera lista ordenada por años. Esta lista puede ordenarse ahora por meses dentro de cada año. Finalmente, se puede poner orden dentro de cada mes en la parte que requiere más permutaciones, que es la de los días.

Utilizando este método, la ordenación se realiza primeramente dentro de todo el grupo y a continuación en partes separadas. En cualquiera de las fases, los elementos están en cierta medida ordenados.

El proceso anterior es más rápido que llamar a una rutina que vaya examinando y ordenando todos los días, después reordene algunos de estos días debido a la influencia de los meses y por último realice una tercera ordenación debida a la influencia de los años.

Se podría utilizar una subrutina adicional para convertir una secuencia de entrada «normal» en una secuencia de fechas «invertida» para llevar a cabo la ordenación.



ORDENACION DE SHELL

La llamada ordenación de **Shell** es una de las dos alternativas más conocidas al método de la burbuja, las cuales han de ser tomadas en consideración cuando el número de elementos a ordenar pasa de unos diez. (Date cuenta que la comparación de una nueva entrada única con una lista ya ordenada, significa que hay que comparar únicamente dos elementos).

La ordenación de **Shell** utiliza una técnica de búsqueda binaria que opera dividiendo sucesivamente en mitades la lista original no ordenada hasta que puede establecerse la posición relativa del nuevo elemento introducido. En cada ocasión interviene en la comparación un nuevo par de valores hasta que se hace una nueva pasada sin permutaciones.

Un módulo de ordenación de **Shell** tiene la forma típica que se muestra a continuación. Añádele esta subrutina a tu programa anterior, modificando el GOSUB que hay en la mitad de la línea 90 para probarla:

Teclea

```
1999 REM METODO DE SHELL
2000 LET Z=AA
2010 IF Z<=1 THEN  RETURN
2020 LET Z=INT (Z/2): LET Y
     =AA-Z
2030 LET ZZ=0
2040 FOR X=1 TO Y
2050 LET XX=X+Z
2060 IF A(X)>A(XX) THEN LET
     YY=A(X): LET A(X)=A(XX)
     : LET A(XX)=YY: LET ZZ=1
2070 NEXT X
2080 IF ZZ>0 THEN GO TO 2030
2090 GO TO 2010
```

¿Cómo funciona este programa? La forma más sencilla de ver lo que sucede es analizar un ejemplo en detalle. Imagínate que se te presenta la siguiente lista de números para su ordenación:

```
ARRIBA:                ABAJO:
67 35 72 19 47 38 11 96
```

En la ordenación de **Shell**, se divide la lista inicialmente en dos partes y se comparan los valores del primer elemento de cada uno de los grupos. Cuando el valor más cercano al principio de la lista es mayor, se efectúa un intercambio. Al dividir en dos grupos se obtienen lo valores iniciales de 67 y 47 que se comparan y se intercambian:

```
ARRIBA:                      ABAJO:
47 35 72 19 : 67 38 11 96
```

Después de un cambio, se compara el siguiente par de valores, que en este caso son 35 y 38. En este caso no se produce intercambio. A continuación se comparan 72 y 11, permutándose. Por último se comparan 19 y 96. La lista queda así:

```
ARRIBA:                      ABAJO:
47 35 11 19 : 67 38 72 96
```

En este punto cada una de las mitades se subdivide de nuevo:

```
ARRIBA:                ABAJO:
47 35:11 19:67 38:72 96
```

El primer valor, que es 37, se compara con el valor que figura en tercer lugar, el 11, intercambiándose con él. El 35, que figura en segundo lugar, se compara con el 19 que está en cuarto

lugar, realizándose una nueva permutación. El 35 queda ahora en la cuarta posición. El valor 47, que se ha sometido con éxito a una comparación, se compara con 67, pero esta vez sin éxito. El cuarto valor, que ahora es 35, se compara con 38 quedándose donde estaba. El quinto valor, 67, se compara con el de la séptima posición, que es el 72, quedando ambos donde estaban. El 38, que figura en sexta posición, se compara sin éxito con 96. En la tabla 2 se muestra la secuencia. Como ocurría antes, se muestran entre paréntesis los valores que intervienen en la comparación: para saber si se ha producido o no un cambio, tienes que observar lo que ocurre en la línea siguiente.

La lista sufre una nueva división para dar los siguientes grupos:

```
ARRIBA:              ABAJO:
11:19:47:35:67:38:72:96
```

Los valores se comparan de nuevo, abriéndose camino los valores más bajos hacia el principio de la lista. Observa que la rutina solamente compara bloques adyacentes. El primero se compara con el segundo, el segundo con el tercero, y así sucesivamente a lo largo de toda la lista, como se observa en la tabla 3.

La ordenación continúa hasta llegar a los valores finales que ya están en orden, volviendo de nuevo al principio, como se observa en la tabla 4.

Aunque la lista ya está ordenada, el programa realizará una nueva pasada. Cualquier valor nuevo puede entonces ser comparado con esta lista ordenada de la misma forma que en el método de la burbuja, ya que eso es esencialmente lo que se hace en cada comparación binaria.

```
TABLA 2     [47]  35 :[11]  19 : 67   38 : 72   96
             11  [35]: 47  [19]: 67   38 : 72   96
             11   19 :[47]  35 :[67]  38 : 72   96
             11   19 : 47  [35]: 67  [38]: 72   96
             11   19 : 47   35 :[67]  38 :[72]  96
             11   19 : 47   35 : 67  [38]: 72  [96]
             11   19 : 47   35 : 67   38 : 72   96
```

```
TABLA 3     [11]:[19]: 47 : 35 : 67 : 38 : 72 : 96
             11 :[19]:[47]: 35 : 67 : 38 : 72 : 96
             11 : 19 :[47]:[35]: 67 : 38 : 72 : 96
             11 : 19 : 35 :[47]:[67]: 38 : 72 : 96
             11 : 19 : 35 : 47 :[67]:[38]: 72 : 96
             11 : 19 : 35 : 47 : 38 :[67]:[72]: 96
```

```
TABLA 4     [11]:[19]: 35 : 47 : 38 : 67 : 72 : 96
             11 :[19]:[35]: 47 : 38 : 67 : 72 : 96
             11 : 19 :[35]:[47]: 38 : 67 : 72 : 96
             11 : 19 : 35 :[47]:[38]: 67 : 72 : 96
             11 : 19 : 35 : 38 :[47]:[67]: 72 : 96
```

ORDENACION DE SHELL-METZNER

La ordenación por el método de **Shell-Metzner** utiliza una rutina de ordenación binaria aún más rápida, que se deriva del algoritmo de **Shell**:

Teclea

```
2999 REM METODO SHELL-METZNER
3000 LET Z=AA
3010 LET Z=INT (Z/2)
3020 IF Z=0 THEN  RETURN
3030 LET Y=AA-Z: LET ZZ=1
3040 LET X=ZZ
3050 LET XX=X+Z
3060 IF A(X)<=A(XX) THEN  GO
     TO 3090
3070 LET W=A(X): LET A(X)=A
     (XX): LET A(XX)=W: LET
     X=X-Z
3080 IF X>=1 THEN GO TO 3050
3090 LET ZZ=ZZ+1
3100 IF ZZ<=Y THEN GO TO 3040
3110 GO TO 3010
```

Añade esta rutina a tu anterior programa de demostración, modificando la instrucción GOSUB de la línea 90 y pruébala.

Como puedes ver, en comparación con el método de la burbuja es extremadamente rápida para el caso en que tengas cincuenta números aleatorios. Pero haz una prueba con conjuntos de números realmente grandes. Mientras trabaja el método de la burbuja puedes irte a tomar un café o a hacer cualquier otra cosa.

DECODIFICACION DE TEXTOS

- DECODIFICADOR EN LENGUAJE ENSAMBLADOR
- COMO SE PRODUCE LA DECODIFICACION
- CONVERSION A BINARIO DEL MATERIAL CODIFICADO

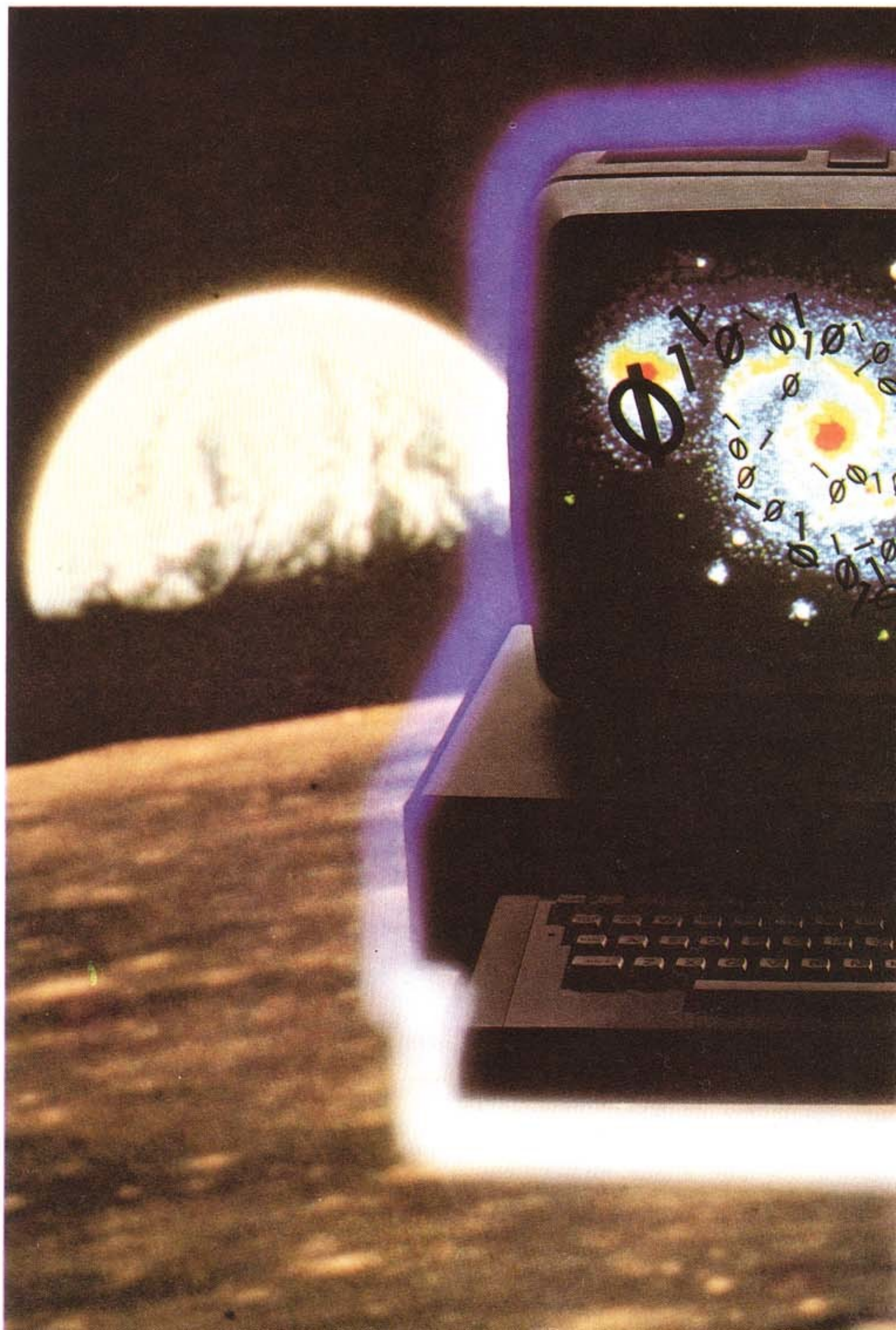
Completamos el compresor de texto añadiendo la rutina de decodificación. Por si no dispones de un ensamblador adecuado, aquí tienes también un listado completo en hexadecimal de todo el programa compresor.

```
FIRST  DEFB CUP-CODE
       DEFB CN-CODE
       DEFB CT-CODE
       DEFB CN-CODE
       DEFB CH-CODE
       DEFB CE-CODE
       DEFB CN-CODE
       DEFB CE-CODE
       DEFB CR-CODE
       DEFB CH-CODE
       DEFB CS-CODE
       DEFB CL-CODE
       DEFB CN-CODE
       DEFB CE-CODE
       DEFB CO-CODE
       DEFB CA-CODE
       DEFB CA-CODE
       DEFB CD-CODE
       DEFB CE-CODE
       DEFB CE-CODE
       DEFB CE-CODE
       DEFB CT-CODE
       DEFB CE-CODE
```

```
        DEFB CE-CODE
        DEFB CE-CODE
        DEFB CU-CODE
        DEFB Cp-CODE
        DEFB CU-CODE
        DEFB CE-CODE
        DEFB CAR-CODE
        DEFB CAR-CODE
        DEFB CSP-CODE
        DEFB CAR-CODE
SECOND  DEFB CT-CODE
        DEFB CT-CODE
        DEFB CE-CODE
        DEFB CF-CODE
        DEFB CI-CODE
        DEFB CO-CODE
        DEFB CT-CODE
        DEFB CI-CODE
        DEFB CD-CODE
        DEFB CO-CODE
        DEFB CT-CODE
        DEFB CN-CODE
        DEFB CS-CODE
        DEFB CA-CODE
        DEFB CE-CODE
        DEFB CH-CODE
        DEFB CO-CODE
        DEFB CT-CODE
        DEFB CL-CODE
        DEFB CH-CODE
        DEFB CI-CODE
        DEFB CO-CODE
        DEFB CI-CODE
        DEFB CA-CODE
        DEFB CI-CODE
        DEFB CX-CODE
        DEFB CT-CODE
        DEFB CO-CODE
        DEFB CI-CODE
        DEFB CUP-CODE
        DEFB CPO-CODE
        DEFB CAR-CODE
        DEFB CUP-CODE
LO      DEFB 0
        DEFB %1
        DEFB %11
        DEFB %111
        DEFB %1111
        DEFB %11111
        DEFB %111111
        DEFB %1111111
UP      DEFB %11111111
        DEFB %11111110
        DEFB %11111100
        DEFB %11111000
```

90 INPUT Juegos

```
        DEFB %11110000
        DEFB %11100000
        DEFB %11000000
        DEFB %10000000
        DEFB 0
ESETUP  LD   HL,(23627)
        LD   D,0
        LD   C,2
EREPEAT     LD      A,(HL)
        CP   #E0
        JR   NC,VFOR
        CP   #C0
        JR   NC,VSA
        CP   #A0
        JR   NC,VMN
        CP   #80
        JR   NC,VNA
        CP   #60
        JR   NC,VN
        JR   VS
VFOR    LD   E,19
        ADD  HL,DE
        JR   EREPEAT
VMN     INC  HL
        LD   A,(HL)
        CP   #EC
```



```
        JR    C,VMN
VN      LD    E,6
        ADD   HL,DE
        JR    EREPEAT
VS      CP    "Z"
        JR    NZ,VSA
        PUSH  HL
        DEC   C
        JR    Z,FINDEX
VSA     INC   HL
        LD    E,(HL)
        INC   HL
        LD    D,(HL)
        INC   HL
        ADD   HL,DE
        LD    D,0
        JR    EREPEAT
VNA     CP    "z"+#20
        JR    NZ,VSA
        PUSH  HL
        POP   IY
        DEC   C
        JR    Z,FINDEX
        JR    VSA
FINDEX  POP   HL
        INC   HL
        RET
```

En el capítulo anterior vimos el método de codificar textos en binario de forma que ocupen menos cantidad de memoria que con la codificación habitual en ASCII.

Los listados en lenguaje ensamblador corresponden a un programa que realiza esta función.

Sin embargo, limitarse únicamente a la codificación es algo que sirve de bien poco; lo único que conseguirías con ello sería encontrarte frente a una incomprensible masa de números binarios. Por eso hace falta un programa que sea capaz de realizar la decodificación. Los listados que siguen a continuación te permitirán añadir al programa la sección decodificadora que te convierta de nuevo el mensaje en un texto legible. También veremos la versión hexadecimal completa del compresor de textos, que te será de utilidad en el caso de que no poseas un ensamblador adecuado. Una vez que hayas completado el programa, consérvalo almacenado en cinta.

En el próximo capítulo se muestra cómo utilizar el programa en un juego de aventuras, así como la forma de desarrollar juegos con el compresor de textos.

FUNCIONAMIENTO

Partiendo desde el principio del texto codificado en binario, la máquina va tomando los bits de uno en uno, literalmente construyendo un número binario cada vez más largo. Después de la adición de cada nuevo bit, el número se compara con los códigos de la tabla de conversión, buscando algún emparejamiento. Si no se encuentra ninguno o, por el contrario, se encuentra más de uno, el programa añade al número un nuevo bit. El procedimiento se repite hasta que se encuentra un emparejamiento exclusivo para el código binario.

Si la lectura del código se inicia en el punto adecuado, inevitablemente se ha de encontrar un emparejamiento único.

Después de esto, el carácter se inserta en una cadena, dejándolo listo para ser utilizado en BASIC. El código binario existente se elimina y se pasa a construir el siguiente número. El ordenador se entera de que el trozo de texto está completo cuando se encuentra con el código de «fin de mensaje»; el programa en BASIC, del que nos ocuparemos en el siguiente capítulo de esta serie, se encargará entonces de visualizar el texto sobre la pantalla.

Si has decidido utilizar el listado en ensamblador, tienes que llamar al código fuente existente para añadirle a continuación la parte de decodificación que presentamos ahora. Después de introducir todo el listado, ensambla el programa completo y almacénalo en cinta o disco, utilizando el procedimiento previsto para el ensamblador empleado.

```
DFIND   PUSH  IX
        PUSH  IY
        CALL  ESETUP
        LD    C,(IY+8)
        LD    B,(IY+9)
        ADD   IY,BC
        LD    C,(HL)
        INC   HL
        LD    B,(HL)
        INC   HL
        PUSH  HL
        ADD   HL,BC
        LD    (DCTEST+1),HL
        LD    A,7
        LD    (BITTY+1),A
        LD    HL,CSP
        LD    (CHAR+2),HL
        POP   HL
        LD    A,H
        JR    PUSHL
```

```
DCHASH LD   A,9
DCLOOP CP   #20
       CALL NC,DECODE
       AND  A
       JR   Z,DCHASH
       POP  HL
       LD   (HL),A
       INC  HL
PUSHL  PUSH HL
DCTEST LD   BC,#FFFF
       SBC  HL,BC
       JR   NZ,DCLOOP
       POP  HL
       POP  IY
       POP  IX
       RET
DECODE LD   IX,LO
BITTY  LD   DE,#00FF
       ADD  IX,DE
       LD   A,(IY)
       XOR  (IY+1)
       AND  (IX+1)
       XOR  (IY+1)
       LD   B,E
       INC  B
BRAN   RRCA
       DJNZ BRAN
       RLA
       DEC  E
       DEC  E
CHAR   LD   IX,#FFFF
       JR   C,LONGD
       RLA
       JR   C,THREE
       LD   A,(IX+FIRST-CODE)
       JR   FOUND
THREE  DEC  E
       RLA
       LD   A,(IX+SECOND-CODE)
       JR   NC,FOUND
       XOR  A
       JR   FOUND
LONGD  RRA
       DEC  E
       DEC  E
       DEC  E
       CP   255+CEIGHT-CLAST
       JR   NC,EIG
       ADD  A,255+CEIGHT-CLAST
       SRA  A
       OR   #80
       CP   255+CSEVEN-CLAST
       JR   NC,SEV
       ADD  A,255+CSEVEN-CLAST
       SRA  A
       CP   255+CSIX-CLAST
       JR   NC,SIX
       ADD  A,255+CSIX-CLAST
       SRA  A
       JR   FIV
EIG    DEC  E
SEV    DEC  E
SIX    DEC  E
FIV    ADD  A,CLAST+1-CODE
FOUND  LD   HL,CODE
       LD   C,A
       ADD  HL,BC
       LD   A,E
       CP   %10000000
       JR   C,NOCHAN
       ADD  A,8
       INC  IY
NOCHAN LD   (BITTY+1),A
       LD   A,(IX)
       CP   "_"
       LD   A,(HL)
       JR   NZ,NOTLWR
       LD   IX,CPUN
       LD   (CHAR+2),IX
       SUB  #20
       RET
NOTLWR CP   "^"
       JR   NZ,NOTUPP
       CALL DECODE
       JR   NZ,NOTUPP
       ADD  A,#20
NOTUPP LD   (CHAR+2),HL
       CP   " "
       JP   Z,DECODE
       CP   " "
       RET  Z
       XOR  #20
       RET
```

Si no tienes ensamblador para el **Spectrum**, puedes utilizar el listado hexadecimal que viene a continuación. Pero antes de introducir el código hexadecimal tienes que ejecutar los siguientes comandos en modo directo:

```
CLEAR 64599
NEW
```

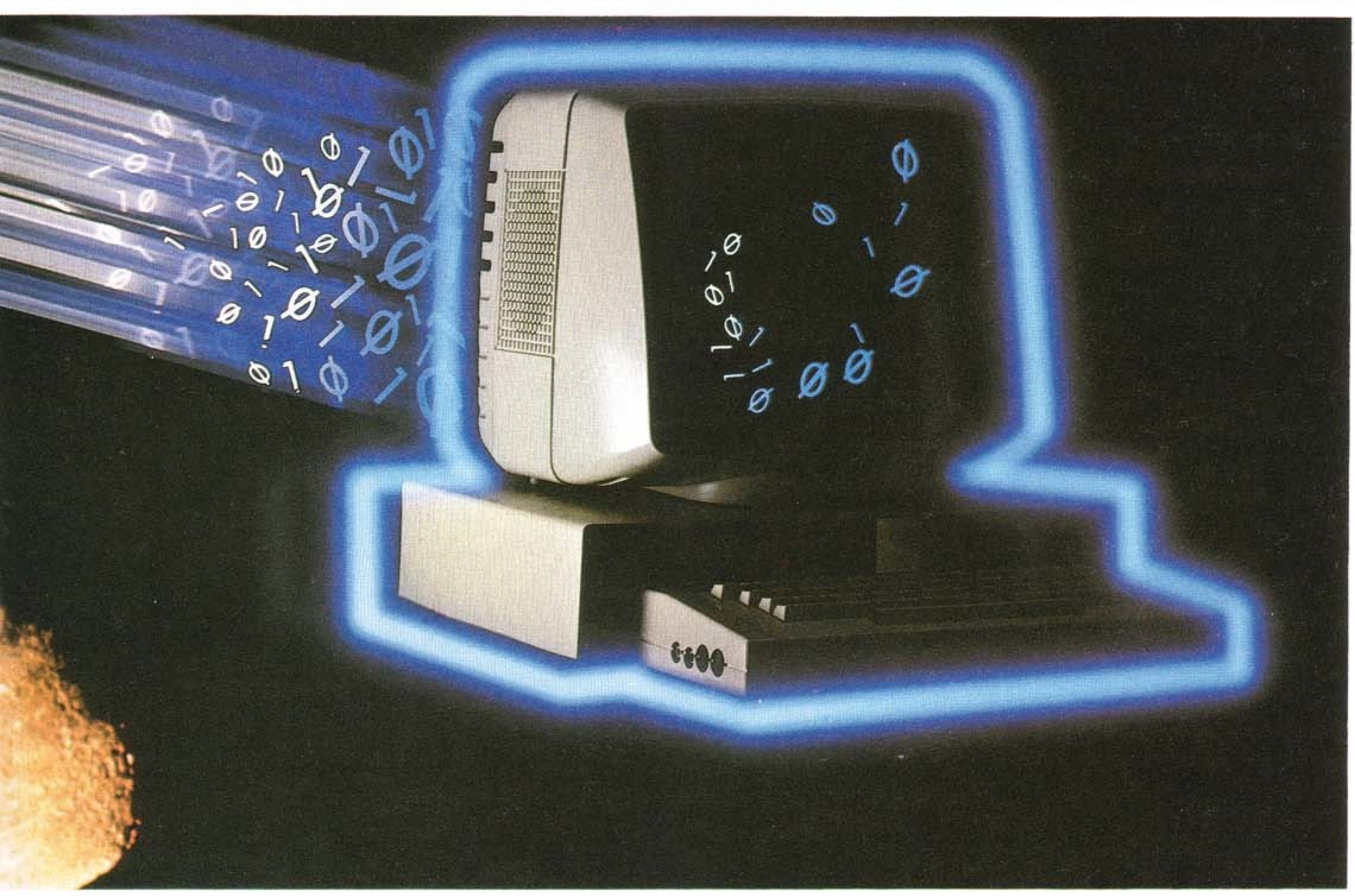
Ahora tienes que transcribir el siguiente programa, el cual has de conservar en cinta para poder introducir el código máquina de la decodificación que se presenta más adelante:

```
10 POKE 23658,8
20 INPUT "DIRECCION COMIENZO?
    ";SA
30 INPUT LINE D$
40 IF D$="" THEN GOTO 30
50 IF D$(1)="#" THEN STOP
60 IF LEN D$=1 THEN GOTO 30
70 LET S$=D$(1 TO 2)
80 FOR N=1 TO 2
90 IF S$(N)>"9" THEN LET
   S$(N)=CHR$(CODE S$(N)-7)
100 IF S$(N)>"?" OR S$(N)<"0"
    THEN PRINT FLASH 1;
    "CARACTER NO VALIDO"
    :GOTO 30
110 NEXT N
120 POKE SA,(CODE S$(1)-48)
    *16+CODE S$(2)-48
130 PRINT SA,D$( TO 2)
140 LET D$(3 TO )
150 LET SA=SA+1
160 GOTO 40
```

Seguidamente teclea RUN e introduce la dirección de comienzo, es decir, 64600. Ahora debes teclear todos los números hexadecimales (pero no

los espacios); cuando hayas terminado conserva en cinta el programa en código máquina, por medio del comando SAVE «txt comp» CODE 64600,684.

```
21 0B 00 22 67 FC C9 DD E5 FD
E5 CD DC FD 01 FF FF FD 71 08
FD 70 09 FD E5 FD 09 4E 23 46
E5 23 09 22 8D FC 3E 07 32 E6
FC 21 69 FD 22 E3 FC E1 23 E5
7E A7 01 FF FF ED 42 28 07 CD
B5 FC C6 00 28 ED E1 3E 00 CD
B5 FC FD 23 FD E5 E1 C1 3A E6
FC C6 F9 ED 42 22 67 FC F1 E1
DD E1 C9 06 20 FE 40 38 02 EE
20 4F 21 88 FD 96 28 1C FE 20
20 0B 3E 7E E5 C5 CD B5 FC C1
E1 18 0D FE E0 20 04 3E 7F 18
EF 79 2B 10 E2 C9 78 DD 21 FF
FF 1E FF 1D 1D 3D 28 13 DD BE
20 20 04 3E 00 18 31 1D DD BE
41 20 09 3E 40 18 27 3E 60 1D
18 22 1D 1D 1D 1D 1D C6 E0 FE
F8 30 17 1C CB 27 C6 08 FE F0
30 0E 1C CB 27 C6 10 FE C8 30
05 1C CB 27 C6 38 4F 7E FE 5E
28 0D DD 7E 00 FE 5F 20 03 21
89 FD 22 E3 FC 79 ED 4B E6 FC
41 04 07 10 FD DD 21 CB FD DD
09 47 DD A6 01 FD B6 00 FD 77
00 78 DD A6 09 FD 77 01 7B FE
80 38 04 C6 08 FD 23 32 E6 FC
3E 00 C9 20 41 53 4F 54 52 49
4C 45 43 5E 5F 55 4D 50 57 59
4E 42 47 44 46 56 48 4B 51 58
4A 5A 5B 5C 5D 0A 11 04 11 17
08 11 08 05 17 02 07 11 08 03
01 C1 14 08 08 08 04 08 08 08
0C 0E 0C 08 0B 0B 00 0B 04 04
08 15 06 03 04 06 14 03 04 11
02 01 08 17 03 04 07 17 06 03
06 01 06 1A 04 03 06 0A 1E 0B
0A 00 01 03 07 0F 1F 3F 7F FF
FE FC F8 F0 E0 C0 80 00 2A 4B
5C 16 00 0E 02 7E FE E0 30 12
FE C0 30 26 FE A0 30 0F FE 80
30 28 FE 60 30 0D 18 10 1E 13
19 18 E4 23 7E FE E0 38 FA 1E
06 19 18 D9 FE 5A 20 04 E5 0D
28 16 23 5E 23 56 23 19 16 00
18 C7 FE 9A 20 F2 E5 FD E1 0D
28 02 18 EA E1 23 C9 DD E5 FD
E5 CD DC FD FD 4E 08 FD 46 09
FD 09 4E 23 46 23 E5 09 22 61
FE 3E 07 32 72 FE 21 69 FD 22
8C FE E1 7C 18 0D 3E 09 FE 20
D4 6D FE A7 28 F6 E1 77 23 E5
01 FF FF ED 42 20 ED E1 FD E1
DD E1 C9 DD 21 CB FD 11 FF 00
DD 19 FD 7E 00 FD AE 01 DD A6
01 FD AE 01 43 04 0F 10 FD 17
1D 1D DD 21 FF FF 38 12 17 38
05 DD 7E 20 18 2F 1D 17 DD 7E
41 30 28 AF 18 25 1F 1D 1D 1D
FE F8 30 18 C6 F8 CB 2F F6 80
FE F4 30 0F C6 F4 CB 2F FE EA
30 08 C6 EA CB 2F 18 03 1D 1D
1D C6 20 21 69 FD 4F 09 7B FE
80 38 04 C6 08 FD 23 32 72 FE
DD 7E 00 FE 5F 7E 20 0B DD 21
89 FD DD 22 8C FE D6 20 C9 FE
5E 20 07 CD 6D FE 20 02 C6 20
22 8C FE FE 5F CA 6D FE FE 20
C8 EE 20 C9
```



UTILIZA TU COMPRESOR DE TEXTOS

- AHORRO DE MEMORIA
- CODIFICACION DE LAS SENTENCIAS PRINT
- LA RUTINA DE DECODIFICACION
- ADAPTACION DEL PROGRAMA DE AVENTURAS

Sácale punta al lápiz y ponte a trabajar en tu obra maestra. Con un compresor de textos que funcione a tope puedes escribir la aventura épica con la que siempre soñaste.

En este momento dispones de un programa en código máquina almacenado en cinta. Tanto si has arrancado a partir del listado en lenguaje en ensamblador, como si lo has hecho partiendo del hexadecimal, el programa es exactamente el mismo. Pero para poder utilizar el código máquina tienes que escribir algo en BASIC.

En esta parte veremos la manera de utilizar el BASIC con el compresor de textos en código máquina y cómo incorporarlo en un juego de aventuras.

DESARROLLO DE JUEGOS DE AVENTURAS

Hemos visto anteriormente la manera de escribir juegos de aventuras. Con unas pequeñas modificaciones de los programas, puedes utilizar el compresor de textos para ahorrar una gran cantidad de memoria.

Para empezar, escribe tu aventura con arreglo a las directrices dadas anteriormente, pero procura incluir mensajes que sean cortos, ya que más adelante los eliminaremos. Por ejemplo, a lo mejor quieres escribir algo como ACABAS DE LLEGAR A UN CLARO DEL BOSQUE. HAY UNA PATA DE PALO PARCIALMENTE OCULTA POR UN MANTEL. En vez de teclear todo esto, utiliza en la sentencia PRINT algo más breve, por ejemplo CLARO. Haz lo mismo

en todas las sentencias PRINT, para que puedas comprobar que tu aventura realmente funciona, depurándola y cerciorándote de que no tiene errores lógicos graves.

Cuando estés satisfecho con el propio juego, puedes pasar a utilizar el compresor de textos y escribir de nuevo las sentencias PRINT.

Lo primero que has de hacer es decidir lo que realmente tienen que decir tus mensajes. Si dispones de una impresora, el trabajo puede resultar más sencillo ya que puedes listar el programa y trabajar sobre el papel. Escribe una lista de mensajes junto con los números de línea correspondientes a cada mensaje. Toda sentencia PRINT que contenga más de dos o tres palabras, te va a merecer la pena codificarla. En los mensajes más cortos es poco o nada el espacio de memoria que se ahorra con la codificación, por lo que te puedes ahorrar el trabajo de hacerla.

FUNCIONAMIENTO

Al codificar el texto, el programa en código máquina almacena la información en binario en una matriz llamada Z. Hay otra matriz, A, que contiene los punteros que permiten a la máquina saber en qué parte de Z comienza cada uno de los mensajes.

Cuando quieras decodificar, es decir, escribir en pantalla alguno de los mensajes, se utilizan los punteros que hay almacenados en A para extraer la sección correcta de información en binario con la que alimentar la rutina de *decodificación*. A medida que va siendo decodificada la información en binario, los caracteres se van cargando en una cadena llamada Z$, que previamente ha sido definida con una longitud igual a la del mensaje más largo. Para que aparezca el mensaje en pantalla, no tienes más que mandar al ordenador que haga PRINT Z$ y suministrarle el número adecuado de A que le diga qué parte del mensaje tiene que cargar en Z$.

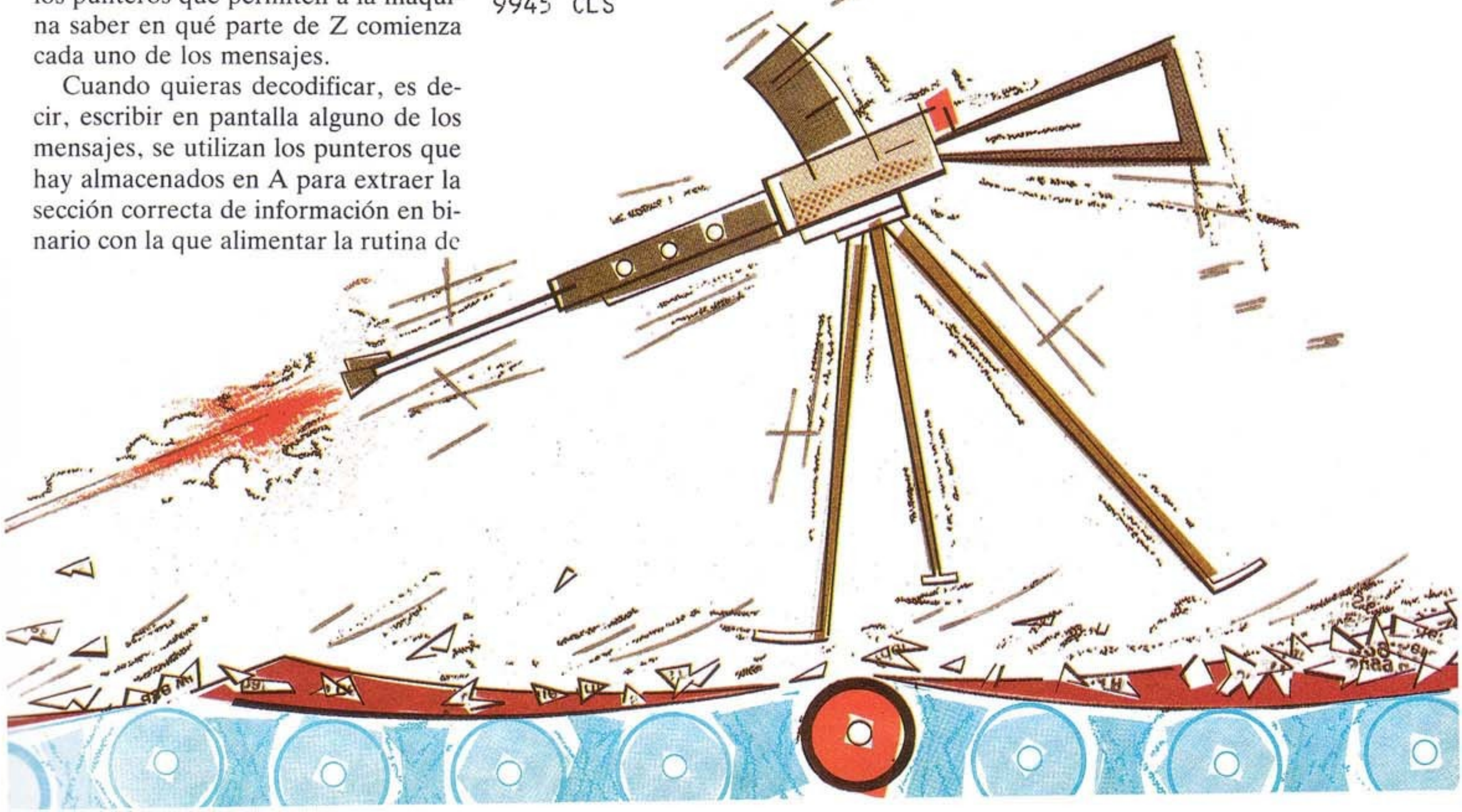
Después de que hayas probado el programa de aventuras con las sentencias PRINT resumidas, puedes empezar a utilizar el compresor de textos.

Con el juego de aventuras todavía cargado en la memoria, carga o teclea las siguientes líneas de programa.

```
9900 CLEAR 64580!:LOAD""CODE
9905 LET MEM=60000!-
     (PEEK 23653+256*PEEK
     23654)
9910 PRINT "ALREDEDOR DE "
     ;MEM;"BYTES LIBRES"
9915 DIM Z(INT(MEM/5))
9920 INPUT "CUANTOS MENSAJES?
     ",N
9925 DIM A(N)
9930 RANDOMIZE USR 64600!
9935 LET Z(1)=11
9940 FOR M=1 TO N
9945 CLS
9950 INPUT INVERSE 1;"TECLEA
     MENSAJE ";(N),LINE Z$
9955 IF Z$="" THEN LET M=N
     :GOTO 9975
9960 RANDOMIZE USR 64607!
9965 LET A(M)=Z(1)
9970 IF MEM+Z(1)<1100 AND MEM
     +Z(1)>900 THEN PRINT
     "ALREDEDOR DE 1000 BYTES
     LIBRES"
9975 NEXT M
9980 PRINT FLASH 1;TAB(6);
     "AHORA GRABA TU PROGRAMA
     ";TAB(6);"NO HAGAS RUN
     O CLEAR!";TAB(31);" "
```

Después de insertar en tu magnetófono la cinta que contiene el compresor de textos, teclea GOTO 9900. El código máquina será cargado en la línea 9900.

El programa examina la cantidad de memoria disponible y te pregunta cuántos mensajes quieres introducir. Cuando le hayas respondido, el programa te invitará a que teclees mensajes uno tras otro. Debes tener cuidado al introducir mensajes que tengan más de una línea de extensión. En tales casos no debes pulsar la tecla de retorno de carro [RETURN] ya que el ordenador lo interpretaría como un final de mensaje. En vez de hacer eso,



llena las líneas con espacios de forma que los mensajes queden adecuadamente formateados; los espacios no ocupan mucha cantidad de memoria.

El programa te avisa cuando quedan unos 1000 bytes en la memoria. Si recibes este mensaje mientras estás tecleando, ten cuidado de que no se te rompa el programa por escribir unas partes sobre otras. Si crees que te vas a pasar, consulta tu lista de mensajes e intenta acortarlos. Después empieza a introducirlos otra vez desde el principio.

Cuando hayas terminado, habrá un mensaje que te indicará que almacenes el programa. Para ello no tienes más que teclear SAVE «nombre del programa», con lo que se almacenarán juntos la aventura y el texto comprimido.

Cuando tengas seguro en cinta el texto comprimido, puedes modificar el programa de aventura para que pueda utilizar el compresor de textos en código máquina junto con los datos almacenados en lugar de las sentencias PRINT provisionales. Borra todas las líneas desde 9900 en adelante, tecleando en su lugar esta subrutina corta:

```
9900 LET Z(1)=A(N):
     RANDOMIZE USR 65067
     :PRINT Z$
9910 RETURN
```

A continuación tienes que ir recorriendo el programa, sustituyendo cada sentencia PRINT correspondiente a un mensaje codificado por una nueva instrucción. Supongamos que quieres llamar al mensaje codificado correspondiente a la segunda sentencia PRINT. Cambia

```
PRINT "MENSAJE"
```

por

```
LET N=2:GOSUB 9900
```

Con esto se le asigna a la variable N un valor igual al número del mensaje y a continuación se inicia la subrutina. Esta subrutina extrae el elemento N-simo de la matriz A, que son los punteros válidos para seleccionar la parte correcta de Z, a continuación carga el mensaje en Z$ y lo imprime.

Finalmente necesitarás un Z$ vacío, para que pueda usarlo el código máquina de forma que pueda presentar el texto decodificado. Define Z$ como quieras, lo más cerca del principio del programa que te sea posible; recuerda que debe tener la longitud suficiente para aceptar el mensaje más largo esperado. Algo del tipo de lo siguiente:

```
1 LET Z$="XXXXXXXXETC"
```

Con la nueva versión del programa así completada, vuelve a almacenarla en cinta. Cuando quieras jugar el juego teclea:

```
CLEAR 64580
LOAD "" CODE (cargando
el compresor de texto)
LOAD "" (aventura
y texto comprimido)
```

Para ejecutar el programa con el texto comprimido, NUNCA pulses RUN, porque perderías todos tus mensajes. Lo que debes hacer es teclear GOTO 1.

RALENTIZADOR DE PROGRAMAS

Un diminuto aparato que cabe en la palma de la mano doblega al poderoso microprocesador del Spectrum, **haciéndole ejecutar a la velocidad que le marque el usuario.**

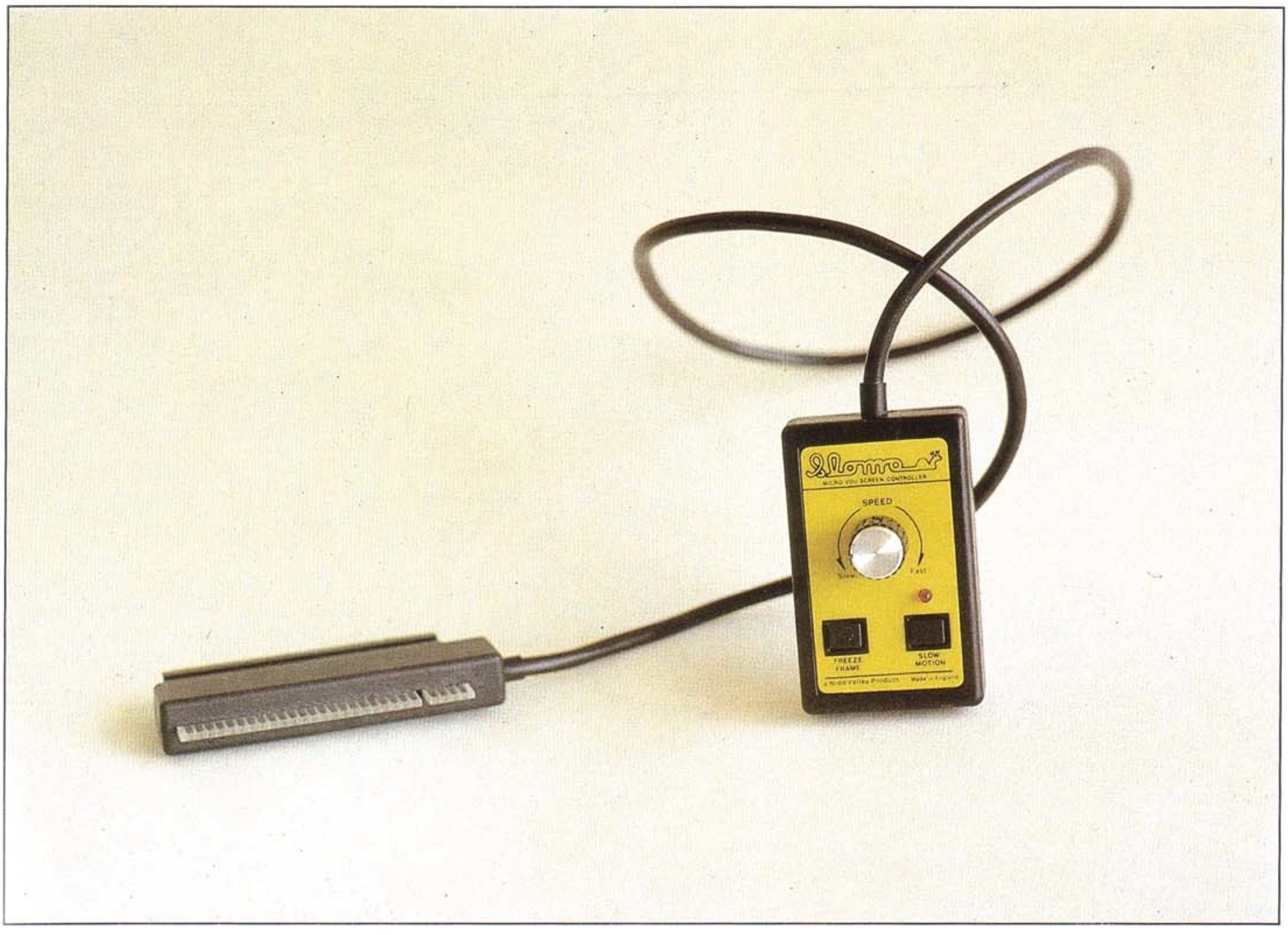
El dispositivo **Slomo** va incorporado en el interior de un receptáculo de plástico negro. De su parte anterior parte un cable que termina en el clásico conector tipo peine, que se insertará en la ranura de expansión del **Spectrum**.

En la cara anterior han sido dispuestos los mandos. Un botón permite controlar la cantidad de giro del cursor de un potenciómetro, entre 0 y 270 grados. Debajo asoma un diminuto LED rojo y dos pulsadores. El papel que juegan estos elementos es lo que veremos seguidamente.

Una vez conectado el **Slomo** al ordenador, se conecta la alimentación y podemos proceder, por ejemplo, a teclear un corto programa. El manual aconseja un bucle de conteo con FOR...NEXT. Al ejecutarlo podemos observar como la ejecución se detiene presionando el pulsador «*freeze*» (congelar). El otro pulsador, «*Slow Motion*» (movimiento lento), permite que entre en acción el control de velocidad de ejecución, mediante el botón giratorio, para que el programa tenga una velocidad totalmente ajustable.

Lo interesante del **Slomo** es su total compatibilidad con cualquier programa, ya que funciona a nivel *hardware*, enviando interrupciones directamente al microprocesador, de tal manera que su control es total, independientemente del tipo de *software* que estemos manejando.

Todos los programas que hemos comprobado en funcionamiento con este dispositivo se han convertido en sumamente dóciles de dominio casi completo de sus evoluciones. Su avance puede contemplarse cómodamente con sólo ajustar la velocidad a nuestro gusto. Esquivar los antaño peligrosos monstruos y misiles se convierte en un juego de niños.



ENTENDIENDO LOS CODIGOS ASCII

- ¿QUE ES EL CODIGO ASCII?
- USO DE NUMEROS EN LUGAR DE CARACTERES
- PROGRAMA PARA IMPRIMIR MENSAJES CIFRADOS

El código ASCII empleado por los ordenadores tiene las suficientes semejanzas para permitirles literalmente «hablar» unos con otros. Este «lenguaje común» tiene también otros usos...

Cada tecla o combinación de teclas de tu ordenador está representada por un código electrónico único. Estos códigos se representan en BASIC por una serie de valores decimales que van desde 0 hasta 255. Por ejemplo, a la letra A le corresponde el valor decimal 65, a la B el 66, etcétera. Cuando tecleas letras o palabras cualesquiera, lo que el ordenador realmente almacena son estos códigos.

Los valores correspondientes a cada configuración de teclas no son los mismos en todos los ordenadores, pero afortunadamente existe al menos una cierta estandarización en la forma en que dichos valores se describen y utilizan.

Este conjunto de valores de designa por las siglas ASCII que significa Código Americano Normalizado para Intercambio de Información (*American Standard Code for Information Interchange*). Este código se elaboró para disponer de un medio por el que los ordenadores pudieran transferirse datos de unos a otros.

Los **Commodore** y **Spectrum** utilizan este código, al menos en parte. El **ZX81**, en cambio, tiene un código completamente diferente, único para esta máquina.

EL CODIGO ASCII

El conjunto completo de caracteres ASCII no es igual para todos los ordenadores. Pero hay bastantes semejanzas. El mayor paralelismo se produce en el intervalo desde 33 hasta 90, que cubre todos los símbolos normales: signos de puntuación, números y letras mayúsculas. Hay una forma muy sencilla de encontrar el código ASCII de un carácter cualquiera; para ello no tienes más que teclear PRINT CODE «X» y te aparecerá el código ASCII de «X» o de cualquier otro carácter. El siguiente programa te permite introducir lo caracteres con más facilidad.

```
30 PRINT "EL CODIGO ASCII
   PARA ";A$;"ES "CODE A$
```

La función inversa de CODE es CHR$, que realiza la conversión del número de código a su correspondiente carácter. El programa que viene a continuación convierte en caracteres todos los códigos ASCII comprendidos entre el 33 y el 90:

```
10 PRINT "CODIGO ASCII",
   "CARACTER"
20 FOR N=33 TO 90
30 PRINT N,CHR$ (N)
40 FOR D=1 TO 500: NEXT D
50 NEXT N
```

Las letras minúsculas del alfabeto se extienden en el margen de 97 a 122, como puedes comprobar cambiando por 122 la última cifra que figura en la línea 20. Si ahora cambias de nuevo dicha cifra por 256, obtendrás la representación del conjunto completo de caracteres. En el manual de tu equipo puedes ver el listado completo de los caracteres.

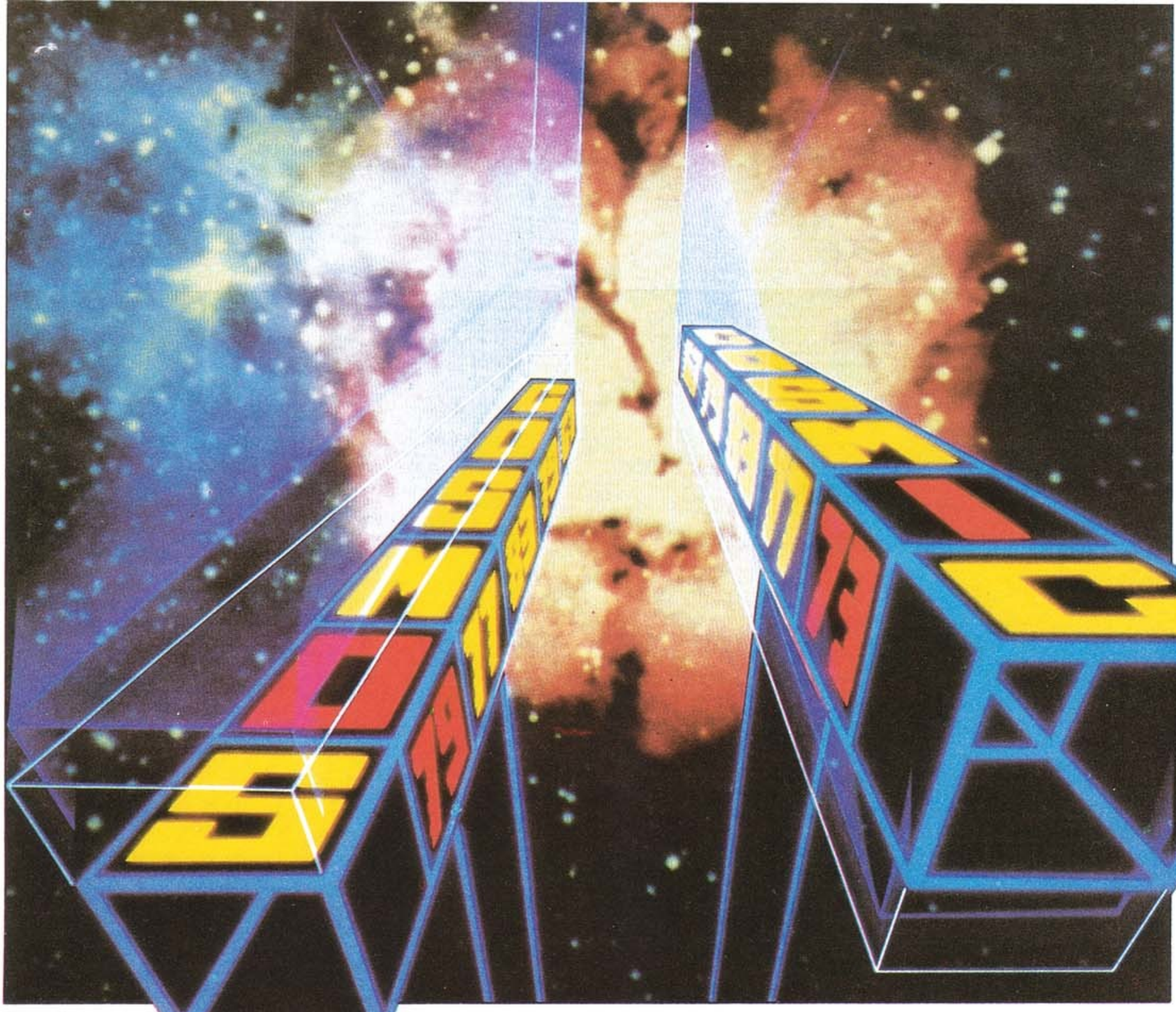
USO DEL CODIGO ASCII

La ventaja de utilizar un número en vez de un carácter, consiste en que puedes modificar matemáticamente dicho número de varias maneras. Si después de hacerlo imprimes el carácter del nuevo número, obtendrás una letra diferente. Esta es la base de muchos programas de escritura de códigos. Se trata de una utilidad de gran valor, ya que no es posible manipular directamente una letra aplicándole una operación matemática.

Los códigos más sencillos se limitan a añadir una cantidad constante a cada número, con lo que resulta de hecho una traslación de la letra en el alfabeto. Por ejemplo, la A se convierte en G, la B en H y así sucesivamente. Pero estos códigos resultan muy fáciles de romper; después de todo, no sería muy difícil escribir un programa de ordenador para probar cada una de las 26 combinaciones posibles, siendo muy fácil detectar de una ojeada cuál es la correcta.

Para que el programa tenga alguna eficacia, tiene que hacer cosas más complicadas con los números; el programa siguiente utiliza un código de forma que cada letra del mensaje se altera de una manera diferente. Este tipo de codificación es muy difícil de romper, a menos que se sepa cuál es la palabra código.

```
10 POKE 23658,8: LET CP=0:
   LET PM=0: LET D$=""
20 CLS
30 INPUT "CUAL ES EL CODIGO?
   ";LINE C$
40 PRINT "ESCRIBE EL MENSAJE:
   -"
50 INPUT LINE M$
60 LET CP=CP+1: IF CP>LEN C$
   THEN LET CP=1
70 LET PM=PM+1
```




```
80 IF PM>LEN M$ THEN GO TO
   200
90 LET F$=M$(PM)
100 IF F$<"A" OR F$>"Z" THEN
    GO TO 150
110 LET F=CODE F$+CODE C$
    (CP)-65
120 IF F>90 THEN  LET F=F-26
130 LET D$=D$+CHR$ F
140 GO TO 60
150 IF F$<"0" OR F$>"9" THEN
    LET D$=D$+F$: GO TO 70
160 LET F=CODE F$+CODE C$
    (CP)-48
170 IF F>57 THEN LET F=F-10:
    GO TO 170
180 LET D$=D$+CHR$ F
190 GO TO 60
200 PRINT "EL MENSAJE
    CODIFICADO ES:-"
210 PRINT 'D$
220 STOP
```

El programa te pide que introduzcas la palabra código y el mensaje, y a continuación imprime la versión cifrada. El mensaje ya está protegido y sólo lo podrá descifrar alguien que conozca la palabra.

La forma en que trabaja el programa es muy sencilla y se parece mucho al programa de codificación simple, con la salvedad de que en vez de sumar una cantidad constante a cada letra, se le suma el código ASCII de la primera letra de la palabra código a la primera letra del mensaje, el de la segunda letra de la palabra código se suma a la segunda letra del mensaje, etcétera. Cuando se llega al final de la palabra código, el ciclo comienza de nuevo.

Si al sumar los números se obtiene un resultado mayor que 90 —lo que supone irse más allá de la letra Z— se restan 26 para mantenerse dentro del margen correspondiente a las letras del alfabeto. Los números reciben un tratamiento separado en las líneas 150 a 180, a fin de mantener sus códigos ASCII entre 48 y 57, que corresponden a los números 0 a 9.

El programa de descifrado es muy parecido al primero; la única diferencia consiste en que se han cambiado unos cuantos +s y −s. La numeración de las líneas es coherente con la del programa anterior, a fin de que puedas almacenar el programa combinado.

Tienes además unas cuantas líneas adicionales que te permiten seleccionar la codificación o decodificación de un mensaje.

```
12 INPUT "(C)ODIFICAR O (D)
   ECODIFICAR?"; LINE A$
14 IF A$="D" THEN GO TO 400
16 IF A$<>"C" THEN GO TO 12
400 CLS
410 INPUT "CUAL EL EL CODIGO
    ? "; LINE C$
420 PRINT "TECLEA EL MENSAJE
    CODIFICADO:-"
430 INPUT LINE M$
440 LET CP=CP+1: IF CP>LEN
    C$ THEN LET CP=1
450 LET PM=PM+1
460 IF PM>LEN M$ THEN GO TO
    580
470 LET F$=M$(PM)
480 IF F$<"A" OR F$>"Z" THEN
    GO TO 530
490 LET F=CODE F$-
    CODE C$(CP)+65
500 IF F<65 THEN LET
    F=F+26
510 LET D$=D$+CHR$ F
520 GO TO 440
530 IF F$<"0" OR F$>"9" THEN
    LET D$=D$+F$: GO TO 450
540 LET F=CODE F$-CODE C$
    (CP)+48
550 IF F<48 THEN LET F=F+10:
    GO TO 550
560 LET D$=D$+CHR$ F
570 GO TO 440
580 PRINT '"EL MENSAJE
    DECODIFICADO ES:-"
590 PRINT 'D$
600 STOP
```

Digamos de paso que si realmente quieres conseguir que tu código sea muy difícil de romper, siempre puedes volver a cifrar tu mensaje cifrado utilizando una segunda palabra código. Acuérdate de colocarlas en el orden correcto cuando descifres de nuevo el mensaje.

COMPARACIONES

Una de las principales aplicaciones de los valores del código ASCII es la comparación de cadenas de caracteres. Dichas comparaciones de cadenas se hacen carácter a carácter y de izquierda a derecha, hasta que se llega al final de la cadena. Supongamos, por ejemplo, que hay que comparar dos cadenas tales como «COSMOS» y «COSMIC». El ordenador primero compara las dos C, después las O, las S y las M. Seguidamente se compara la O de la primera cadena con el correspondiente carácter de la otra cadena, que es la I.

Lo que se compara no es un conjunto arbitrario de valores alfabéticos, sino el valor ASCII de cada carácter. El valor ASCII de la O es mayor que el de la I, por lo que en esta comparación «COSMOS» tiene un valor mayor que «COSMIC». Observa que no se utilizan las sumas de los valores ASCII de las letras para determinar cuál es la cadena que tiene un mayor valor. Cada letra se compara individualmente.

Aquí tienes como ejemplo algunas comparaciones que muestran posibles valores de A$ y B$, junto con los códigos ASCII de los caracteres que intervienen. Es importante comparar los valores uno a uno, el primer carácter de A$ con el primero de B$, siguiendo así en toda la extensión de la palabra más corta, ver tabla 1.

A$/ASCII		B$/ASCII		RELACION
ABC	65,66,67	ABC	65,66,67	A$=B$
ABD	65,66,68	ABCD	65,66,67	A$>B$
ABD	65,66,68	ABCD	65,66,67	A$<>B$
ABC	65,66,67	Abc	65,97,98	A$<B$
COSMI	67,79,83, 77,73	COSMO	67,79,83, 77,79	A$<B$
$1	36,49	$1.0	36,49,46	A$<B$

Como puedes observar en los ejemplos segundo y tercero, se puede escribir la relación de más de una forma. Fíjate también en el último ejemplo, en el que a igualdad de todo lo demás, se considera que es mayor la cadena de caracteres más larga.

COMPROBACION DE LAS ENTRADAS

La función CODE actúa únicamente sobre el primer carácter de la cadena. Así, la instrucción PRINT CODE «USA» dará como resultado el número 85, que es el código ASCII de la letra «U». Este hecho resulta muy útil en la comprobación de las entradas que se suministran a los programas por medio del comando INPUT. Por ejemplo, en el anterior programa de escritura de códigos, la línea 12 te pedía que introdujeras la letra C para cifrar o la letra D para descifrar. A continuación, las dos líneas siguientes encaminan el programa a la sección adecuada. Pero si quieres impedir que alguien escriba «cifrar» o «descifrar» *in extenso*, puedes volver a escribir así las líneas 14 y 16:

```
14 IF CODE A$=68 THEN GO TO
   400
16 IF CODE A$<>67 THEN GO TO
   12
```

CODIGOS DE CONTROL

Algunos códigos ASCII no van asociados a ningún carácter. Por ejemplo, cuando pulsas la tecla ENTER el ordenador almacena el valor 13, pero en lugar de imprimir un carácter en la pantalla el cursor se desplaza al principio de la siguiente línea o se lista el programa. Se dice que el código 13 es el código del retorno de carro, y en algunos casos el código de línea nueva. Teclea PRINT «A»;CHR$(13);«B» y observarás que la «B» se imprime en una nueva línea directamente debajo de la «A».

Aquí tienes la manera de encontrar los códigos ASCII de las teclas que no corresponden a caracteres:

```
10 PRINT CODE INKEY$
20 GO TO 10
```

Prueba a pulsar ENTER, DELETE, o cualquier otra tecla de tu ordenador. Puedes utilizar este método para detectar las pulsaciones de tecla en los programas de juegos. Por desgracia no puedes utilizar los valores de forma inversa, es decir, aunque el código 10 corresponde al movimiento del cursor hacia abajo, PRINT CHR$(10) no hará que el cursor se mueva hacia abajo.

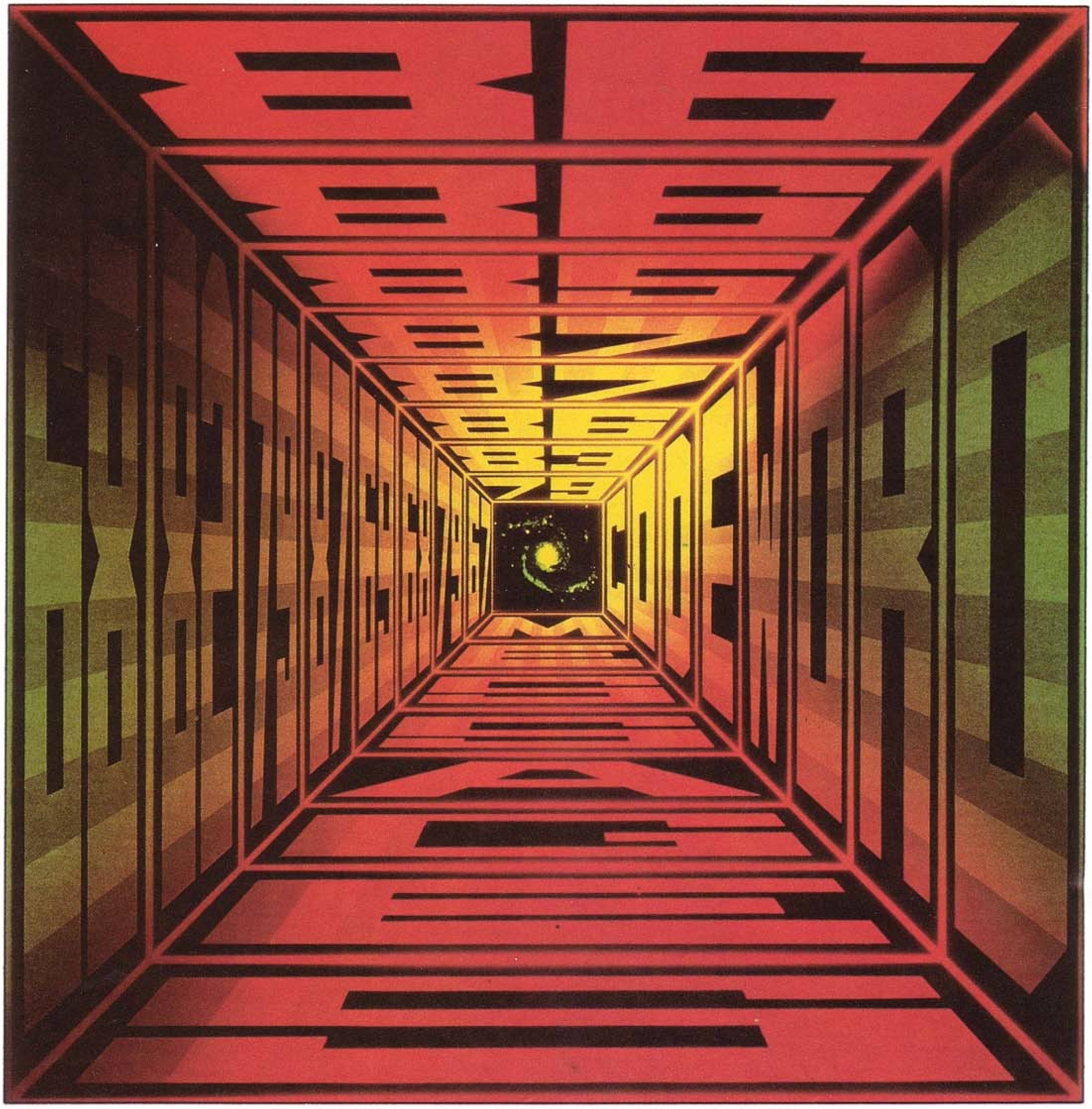
Unos ordenadores utilizan estos códigos más que otros. Hay muchos números disponibles: desde el 1 al 31 y

unos cuantos por encima del 90, pero inicialmente fueron definidos en los días en que los ordenadores estaban conectados a impresoras, por lo que los códigos están relacionados fundamentalmente con el control de aquellas vetustas impresoras. Naturalmente en la actualidad los ordenadores están diseñados para ser utilizados con un monitor de video, por lo que la mayoría de los códigos ya no se aplican, si bien algunos de ellos se pueden utilizar en las impresoras modernas.

El **Spectrum** tiene redefinidos muchos de los códigos para que funcionen sobre la pantalla de un televisor. Por ejemplo, PAPER tiene el código 17 mientras que INK (tinta) tiene el código 16. Así, para imprimir la palabra «TITULO» en la pantalla, con letras rojas sobre fondo verde, puedes utilizar este programa:

```
10 LET A$=CHR$ 17+CHR$
   4+ CHR$ 16+CHR$ 2+
   "TITULO"
20 PRINT A$
```

Cuando se trate de una cabecera que quieras utilizar varias veces en un programa, no tienes más que definir A$ una vez al principio, y a continuación utilizar A$ cada vez que necesites hacerlo.



¡PARTICIPA EN INPUT!

BASES

PROGRAMAS: Una vez desarrollado tu programa, que debe ser original y no haber sido enviado a ninguna otra publicación, puedes enviárnoslo aquí grabado en cassette, diskette o microdrive. Es preferible que vaya acompañado por un listado de impresora, pero no es imprescindible.

El programa habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada: nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

ARTICULOS E IDEAS: Se aplica lo anteriormente dicho para los textos que acompañan a los programas: es decir, conviene detallar al máximo lo que desees que aparezca publicado en la revista, de la manera que te gustaría que otra persona hubiera explicado eso mismo.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvidéis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante cada mes SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas. en material microinformático a elegir por los afortunados.**

¡No os desaniméis!, por muy simples o complejas que puedan parecer vuestras ideas, todas serán revisadas con el máximo interés.

INPUT SINCLAIR
Alberto Alcocer, 46, 4.° B
28016 Madrid

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.

MANEJO DE INTERRUPCIONES

- INTERRUPCIONES EN EL Z80
- LA LINEA INT
- EL COMANDO BEEP Y LAS INTERRUPCIONES
- PROGRAMA MUSICAL

De las 40 patillas que tiene el microprocesador del Spectrum (Z80) **hay dos ligadas directamente con las interrupciones y son INT (pin 16) y NMI (pin 17).**

Al hablar de interrupciones nos referimos a una suspensión temporal de la tarea que en ese momento esté realizando el ordenador para efectuar otra que se le encomiende, retornando luego a la situación anterior.

Se produce una interrupción cuando se activa la línea INT o la NMI. En el primer caso se dice que se ha producido una interrupción «enmascarable» y en el segundo una interrupción «no-enmascarable».

Hay que aclarar que el **Spectrum** no utiliza todas las posibilidades que ofrece el **Z-80** en cuanto a interrupciones. Sólo nos referiremos, pues, a lo que nos afecte.

Hay que matizar que las interrupciones, en sí, no producen directamente más que el efecto de «llamar la atención» del microprocesador. Cuando se detecta una interrupción detiene el programa que está ejecutando y acude a cierto lugar para recibir instrucciones (rutinas de manejo de interrupciones) y ejecutarlos, continuando después con el programa principal en el lugar en que lo dejó. Se dice que las interrupciones son «vectorizadas» ya que dependiendo del modo en que se trabaje existe una dirección de memoria (vector de interrupción) que indica una nueva dirección que corresponde a la rutina a ejecutar.

Una interrupción es «enmascarable» si puede no tenerse en cuenta. Por el contrario, a una interrupción no-enmascarable, siempre hay que hacerle caso. Vamos a poner un ejemplo sencillo para aclarar un poco más los conceptos.

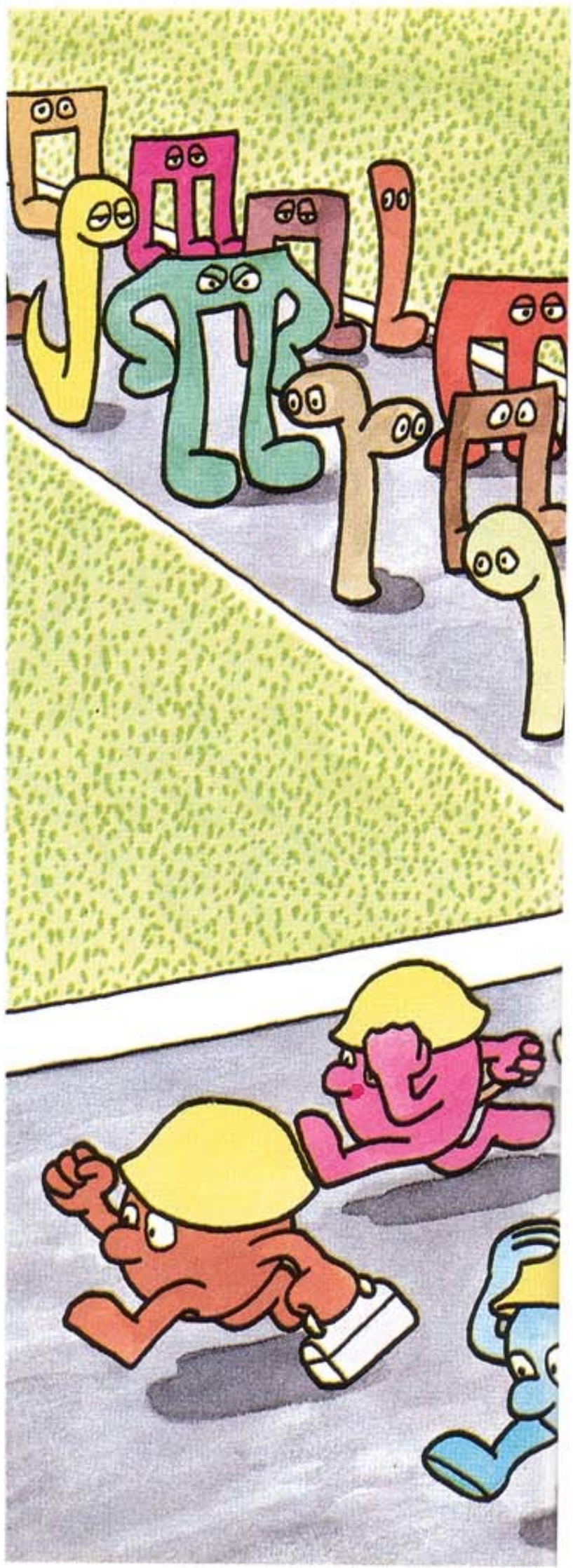
Imaginémonos en una oficina donde existen dos timbres: el del teléfono y el de la alarma de incendio. Estamos trabajando normalmente y en un cierto momento nos interrumpe el sonido de un timbre. Si suena el teléfono, lo normal es dejar lo que estamos haciendo, descolgamos, atendamos la llamada (rutina de contestación) y continuemos con nuestro trabajo hasta que tengamos otra interrupción telefónica. No obstante, si estamos haciendo algo que no nos interesa dejar podemos optar por no coger el teléfono (dejarlo descolgado o desconectar el timbre el tiempo necesario). En este caso estamos enmascarando, ignorando o inhabilitando la llamada. Por supuesto no debemos olvidarnos habilitar de nuevo la situación porque en caso contrario quedaríamos desconectados incapacitados para recibir llamadas del mundo exterior.

Si el timbre que suena, es el de la alarma de fuego, nos guste o no deberemos dejar lo que estemos haciendo y ejecutar las instrucciones oportunas (rutina de instrucción de no-enmascarable) como puede ser salir por la escalera de incendios asignada a esa zona. A ningún arquitecto se le ocurriría poner un interruptor para que en caso de estar muy ocupados no nos moleste ni la alarma de fuego. Este tipo de interrucpiones son por tanto No-enmascarables.

Como programadores en código máquina tenemos sólo a nuestra disposición las interrupciones que nos llegan por la línea INT. Las interrupciones que proceden de la línea NMI (No-enmascarables) tienen prioridad sobre las INT y en el **Spectrum** las controla únicamente el ordenador. Podemos habilitar la interrupciónn mediante la instrucción EI (*Enable Interrupt*)(hexadecimal FB, decimal 251) o inhibirla mediante DI (*Disable Interrupt*)(hexadecimal F3, decimal 243).

Existen tres modos de funcionamiento con interrupciones enmascarables: 0, 1 y 2. El sistema operativo del **Spectrum** utiliza sólo el Modo 1.

Al conectar el ordenador se arranca en Modo 0 (IM0), se ejecuta una rutina de inicialización situada en la ROM y a continuación pasa automáticamente al Modo 1 (IM1) donde permanece indefinidamente durante el funcionamiento estándar.



El modo 1 es el habitual del **Spectrum** y en él, cada vez que se produce una interrucpión interna, siempre que estén habilitadas las interrucpiones (EI), se ejecutan las rutinas de exploración del teclado (RST #38) y un incremento de la base de tiempos. El ritmo a que se producen interrupciones viene fijado por un reloj interno que da impulsos de sincronización cada 20 milisegundos (50 ciclos por segundo). Tanto el teclado como el reloj de tiempo real se dice que «están controlados por interrupciones».

Durante la exploración del teclado, la tecla pulsada se almacena en la variable del sitema LAST KEY (pos. 23561) y se incrementa en uno el valor de la variable del sistema FRAMES (pos. 23672 / 23673 y 23674).

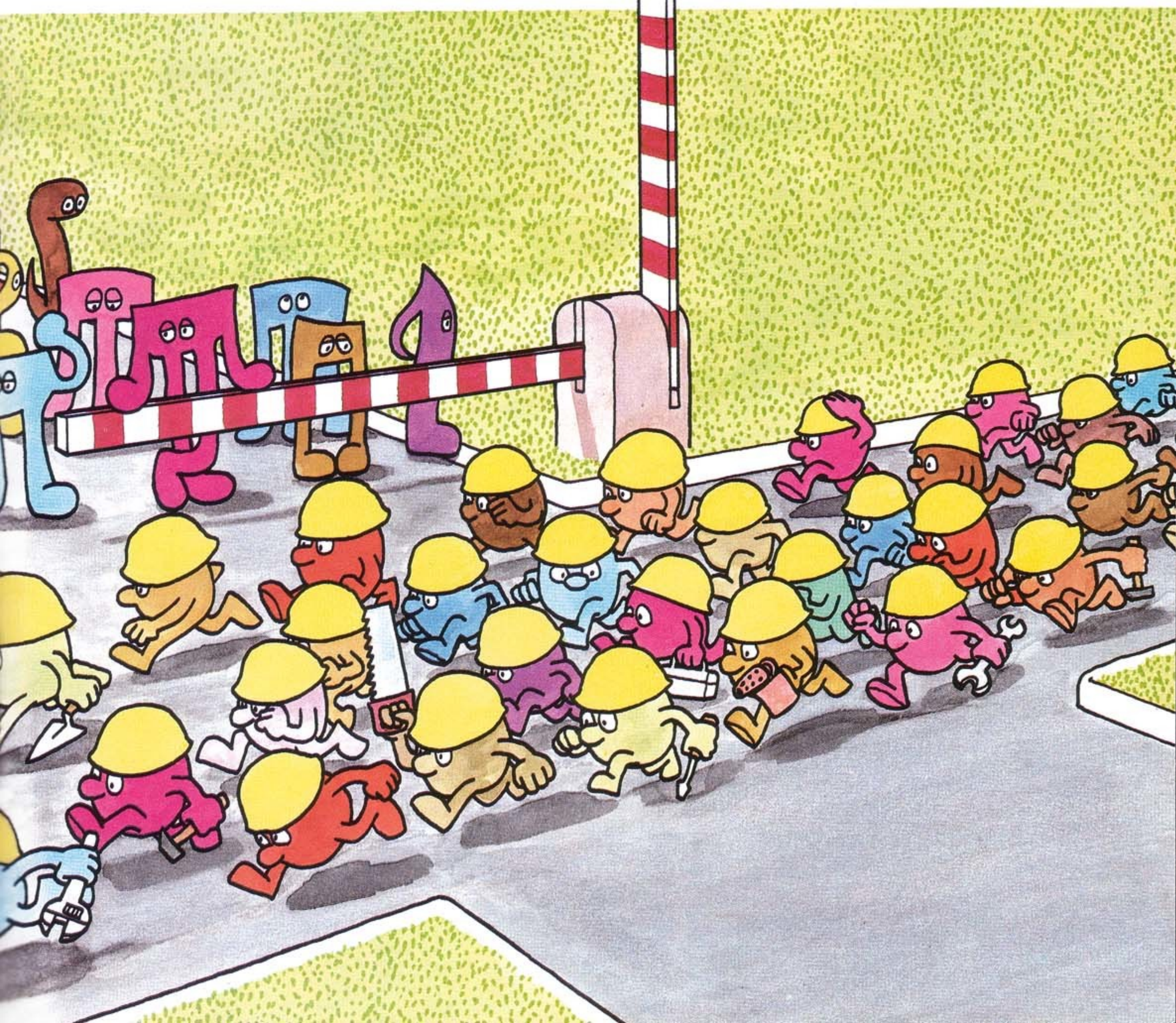
Durante la inhibición de interrupciones (DI) el teclado y el reloj de tiempo real (no confundir con el «reloj» que proporciona los impulsos de sincronización) están bloqueados. O dicho de otra manera el teclado no funciona y el reloj atrasa.

Existe una instrucción RETI para finalizar una interrupción enmascarable retornando a la misma situación inicial por lo que respecta al estado de interrupciones. Existe una equivalente (RETN) para el caso de interrupciones no-enmascarables. La instrucción RET retorna de cualquier rutina de interrupción (ISR), pero sin mantener los valores actuales de estado de interrupciones.

En el **Spectrum** se desactivan las interrupciones (DI) cuando se ejecutan ciertos comandos, por ejemplo, BEEP x,y reactivándose (EI) automáticamente al finalizar. Situaciones análogas se producen durante el paso del contenido del *buffer* de impresora a ésta, durante PRINT o LPRINT etc.

Si en un programa de código máquina no necesitamos actualizar el reloj de tiempo real ni utilizar el teclado podemos inhibir las interrupciones (DI) y el programa rodará más deprisa.

El modo 2 no lo utiliza el **Spectrum**



de forma estándar, pero es el más potente y podemos aprovecharlo en nuestros programas en código máquina.

Cuando está activado el modo 2 (IM2) y se produce una interrupción, se suspende la actividad que se esté ejecutando y el ordenador ejecuta la rutina cuya dirección (vector) es el valor reflejado por el contenido del registro I (byte alto) y el bus de datos (byte bajo). Si no hay ningún periférico conectado, o lo hay pero no ha producido alguna interrupción, tiene siempre el valor 255 (FF en hexadecimal) en el bus de datos. En cualquier otro caso aparecerá el número correspondiente a la interrupción producida.

Vamos a aclarar las cosas con un ejemplo. Supongamos que se ha producido una interrupción en Modo 2, que no existe ningún periférico conectado y que hemos almacenado en el registro I del **Z80** el valor $7D (125 decimal). El ordenador leerá la dirección de ISR (*Interrupt Service Routine*), $7D00 (32000 decimal). El vector de interrupción será pues en este caso 32000. En esta posición (y siguiente) habremos almacenado previamente la posición donde estará situada la rutina que queremos efectuar y desde la cual evidentemente podremos llamar a otras de forma encadenada.

La dirección donde apunta el vector de interrupciones evidentemente no puede ser cualquiera si queremos conseguir un resultado coherente. Por ello el valor que dispogamos en el registro I no puede estar entre $00 y $3F (ROM), $40 y $5C (memoria de pantalla, *buffer* de impresora, variables del sistema, etc.), o cualquier otra parte ocupada por el BASIC, programas en código máquina, etc.

Como ejemplo práctico de manejo de interrupciones analizaremos el siguiente programa mediante el cual conseguiremos tener una música de fondo, hagamos lo que hagamos, lo cual puede ser agradable mientras tecleamos programas o los ejecutamos. Podemos componer la música a nuestro gusto y evidentemente no será de gran calidad, ya que sólo puede oirse «a golpe de interrupciones» pero nos dará la sensación de que todo se produce simultáneamente.

En primer lugar comentaremos el programa en código máquina en sí y para ello nos valdremos del programa fuente el cual aparece a continuación, el cual tiene los siguientes módulos:

- puesta en marcha música de fondo (190-320)
- parada música de fondo (330-360)
- ejecución de notas musicales (370-700)
- inicialización (70-180)
- archivo parámetros (20-60)

El rodaje del programa se comienza con RANDOMIZE USR 64531 (FC13 hexadecimal). Se almacenan las posiciones INTERR y INTERR+1 en las pos. etiqueta VECTOR y se carga el valor 251 en el registro I. A continuación se fija el Modo 2 de interrupciones y se cargan las variables CAMBIO y DURACION con los valores 1 y 0 respectivamente, retornándose al control del BASIC después de habilitar las interrupciones.

Tan pronto se produzca una interrupción se detendrá la ejecución del programa, tal y como se indicó anteriormente ejecutándose la rutina fijada por el vector 64511 en nuestro caso (FBFF, 251 ($FB) del valor de I y 255 ($FF) del valor del bus de datos al no haber conectado ningún periférico. Si hubiera alguno activado evidentemente la rutina no rodaría, pues el vector de interrupción VECTOR apuntaría a otra dirección distinta a INTERR). Desde esta rutina se llama a la ARRANQ para la ejecución de una nota y luego a la rutina de la ROM RST0038h (JP 56 decimal) para la lectura del teclado y actualización del reloj de tiempo real (FRAMES). Finalmente se retorna al BASIC en espera de otra interrupción.

No descenderemos al detalle pero indicamos que la ejecución de la música se consigue básicamente haciendo uso de la rutina BEEPER de la ROM (CALL 949) y las cosas están dispuestas de forma que se detecta el final de los compases que estamos utilizando y se comienza de nuevo, a fin de que la fuente de la música de fondo no tenga fin. El lugar donde tenemos compiladas nuestras notas se fija externamente mediante POKE a las posiciones de la etiqueta ORIG-D.

Si la música nos pone nerivosos y necesitamos concentración la cosa es fácil. Basta ejecutar RANDOMIZE USR 64561 y cesará. Con esta rutina hemos cambiado al Modo 1 en donde sólo se ejecuta RST #38.

Para mayor facilidad ofrecemos un programa en BASIC mediante el cual podemos escribir nuestra propia música y comprobar en todo momento como suena y compilarla si nos gusta para poder rodarla como música de fondo (compartiendo nuestros programas BASIC, en fase de edición o ejecución). Se incluye igualmente un cargador de código máquina con los bytes correspondientes (DATA 4500) para aquellos que no dispongan de uno y tampoco deseen hacer ellos mismos la compilación (nosotros hemos utilizado el GENS3).

Creemos que el programa está suficientemente documentado con los abundantes REM y no requiere ninguna aclaración especial.

Como medida precautoria recordamos la conveniencia de salvar el programa antes de adentrarnos en la aventura musical. Y una vez salvado, al ejecutarlo, hay que tener en cuenta que no se puede utilizar la opción C sin haber ejecutado antes la B.

```
1 REM ** MUSICA BAJO
  INTERRUPCIONES **
5 REM
6 REM INTRODUZCA AQUI SU
  MELODIA EN PARAMETROS DE
  BEEP (T,F)
7 REM
10 DATA 1,12,2,9,1,9,1,9,1,
   8,1,9,2,17,1,12,2,12,1,9,
   2,10,1,10,2,10,1,12,5,14
20 DATA 1,14,2,7,1,7,1,7,1,6
   ,1,7,2,16,1,14,2,14,1,10,
   2,9,1,9,2,9,1,10,5,12
30 DATA 1,12,2,9,1,9,1,9,1,8
   ,1,9,2,17,1,12,2,12,1,12,
   2,11,1,19,2,19,1,19,5,19
40 DATA 1,17,2,16,1,19,1,19
   ,1,18,1,19,2,14,1,19,1,19
   ,1,18,1,19,2,12,1,11,2,12
   ,1,11,3,12
50 DATA 3,10,1,9,1,8,1,9,2,
```

```
   14,1,12,3.2,9,3.2,5,3.2,2
   ,3.2,7,5,5
60 DATA 1,5,1,7,1,9,1,10,2,
   16,1,14,3.2,12,3.2,17,3.2
   ,16,3.2,14,5,12
70 DATA 1,12,2,14,1,14,1,14,
   1,13,1,14,3,16,3,9
80 DATA 2,17,1,17,1,19,1,17,
   1,19,5,21
90 DATA 1,21,2,19,1,17,2,14,
   1,10,3.2,9,3.2,5,3.2,7,3.
   2,4,5,5
96 REM
98 REM INDICADOR FIN MUSICA
   (NO ELIMINAR)
99 DATA 255,255
1000 REM
1001 REM MENU
1002 REM
1005 BORDER 0: PAPER 0: INK
     7: BRIGHT 1: CLS
1010 PRINT AT 3,2; PAPER 2;"
     [28*espacio]"
1020 PRINT AT 4,2; PAPER 2;"
     MUSICA BAJO
     INTERRUPCIONES "
1030 PRINT AT 5,2; PAPER 2;"
     [28*espacio]"
1040 PRINT AT 6,2; PAPER 2;"
     [6*espacio]INPUT
     SINCLAIR[7*espacio]"
1050 PRINT AT 7,2; PAPER 2;"
     [28*espacio]"
1060 PRINT AT 12,0;"A -
     ESCUCHAR LA MELODIA EN
     BASIC";AT 14,0;"B -
     COMPILA LAS NOTAS";AT
     16,0;"C - EJECUCION "
     "MUSICA DE FONDO""""
1080 LET A$=INKEY$
1090 IF A$="A" OR A$="A"
     THEN GO TO 3000
```

```
1100 IF A$="B" OR A$="B"
     THEN  GO TO 4000
1200 IF A$="C" OR A$="C"
     THEN  GO TO 6000
1300 GO TO 1080
3000 REM
3001 REM VERIFICACION DE LA
     MUSICA
3002 REM
3005 RESTORE 10
3010 READ D,F: IF D=255 AND
     F=255 THEN  GO TO 1000
3020 BEEP D/10,F-6: GO TO
     3010
4000 REM
4001 REM CARGA DE LA RUTINA
4002 REM
4005 RESTORE 4500
4010 CLEAR 64504
4015 PRINT AT 10,2; PAPER 1;
     INK 6; FLASH 1;"CODIGO
     DE MAQUINA CARGANDOSE"
4020 FOR N=64505 TO 64639
4030 READ A: POKE N,A
4050 NEXT N
4500 DATA 128,252,0,0,0,0,0,
     0,221,229,229,197,213,
     245,195,54,252,241,209,
     193,225,221,225,195,56,
     0,33,1,252,34,255,251,
     243,62,251,237,71,237,
     94,42,249,251,34,251,
     251,62,1,50,253,251,61,
     50,254,251,251,201,243,
     237,86,251,201,58,253,
     251,254,0,202,10,252,58
     ,254,251,254,0,202,77,
     252,61,50,254,251,195,
     10,252,42,251,251,126,
     254,255,202,110,252,50
     ,254,251,35,126,35,95,
     126,35,34,251,251,103,
     107,17,4,0,205,181,3,
     243,195,10,252,42,249,
     251,34,251,251,62,1,50,
     253,251,61,50,254,251,
     195,10,252
5000 REM
5001 REM COMPILACION DE LA
     MUSICA ELEGIDA
5002 REM
5005 RESTORE 10: CLS : LET
     ORIGD=64640
5008 LET NOTA=0: LET D=ORIGD
5010 READ DURACION,TONO
5015 LET DURACION=INT
     (DURACION*6)
5020 LET FRECUENCIA=(1.05946
     31^TONO)*256
5030 LET PITIDO=INT ((437500
     /FRECUENCIA)-30.125)
5035 IF DURACION=255*6 THEN
     POKE D,255: GO TO 5100
5037 LET NOTA=NOTA+1: PRINT
     AT 10,10;"NOTA ";NOTA
5040 POKE D,DURACION
5060 POKE D+1,PITIDO-(INT
     (PITIDO/256)*256)
5070 POKE D+2,INT (PITIDO/
     256)
5080 LET D=D+3
5090 GO TO 5010
5100 PAUSE 50: CLS : LET
     LONG=(D+2)-ORIGD
5110 PRINT AT 10,2; PAPER 1;
     " LAS NOTAS OCUPAN ";
     PAPER 2;LONG; PAPER 1;
     " BYTES"
5120 PRINT AT 20,5;"PULSE
     UNA TECLA": PAUSE 0: GO
     TO 1000
6000 REM
6001 REM EJECUCION PROGRAMA
     INTERRUPCIONES
6002 REM
6005 RANDOMIZE USR 64531: GO
     TO 1000
```

EL ZOCO DE INPUT

Todo se compra y se vende. Los antiguos zocos fueron lugares destinados a todo tipo de transacciones. INPUT también tiene el suyo. Vuestras operaciones de compra, cambio o venta serán publicadas en esta sección, pero dos son las limitaciones que imponemos:

a) La propuesta tendrá que ver con la microinformática.
b) Nos reservamos el derecho de no publicar aquellos insertos de los que se sospeche un trasfondo lucrativo.

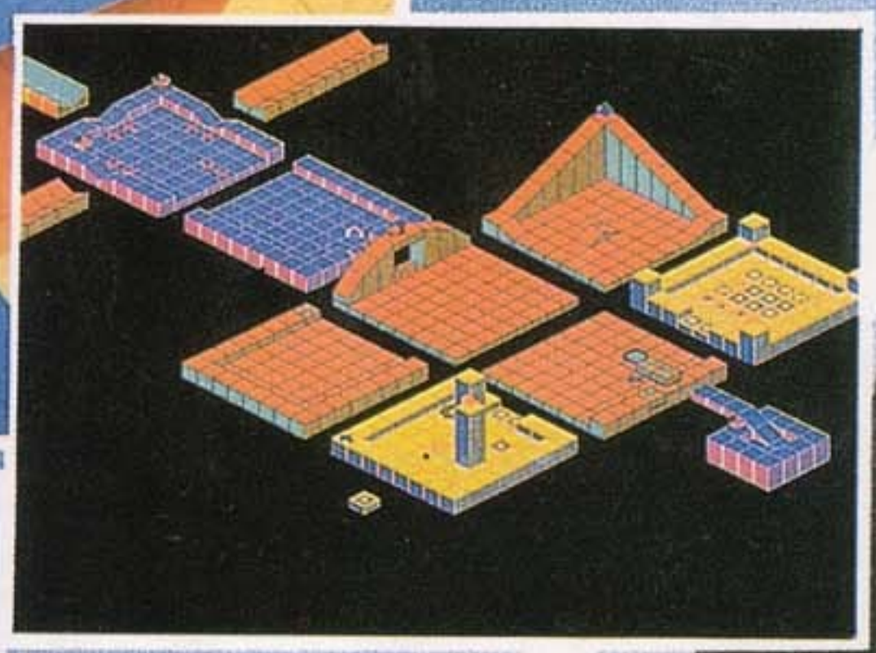
Ahora un ruego. Tratar de resumir al máximo el texto; escribir casi como un telegrama siendo claros y concisos.

Envía tu mensaje a:

INPUT SINCLAIR-ZOCO
c./ Alberto Alcocer, 46
28016 **MADRID**

LO MEJOR PARA SPECTRUM

Spindizzy



SI JUEGAS CON SPINDIZZY ¡PREMIO SEGURO!

...SOLO CON ESCRIBIRNOS DICIENDO CUANTOS SEGUNDOS TE DESCUENTAN CUANDO LA PEONZA CAE AL VACIO.

Si la respuesta es correcta, recibirás gratis en tu domicilio el próximo número de esta revista, y ¡además! entre las cien primeras cartas recibidas sorteamos *¡cinco magníficos compu-robot!* Escríbenos acompañando las instrucciones que te devolvemos con el premio. ¡Ojo, que no valen las fotocopias! El concurso acaba el 30 de septiembre. Pon tu nombre y dirección y el nombre de la revista.

TAMBIEN DISPONIBLE
PARA AMSTRAD Y COMMODORE

Electric Dreams
SOFTWARE

EN TIENDAS ESPECIALIZADAS Y GRANDES ALMACENES, O DIRECTAMENTE POR CORREO O TELEFONO A:

PROEIN, S.A.
Velázquez, 10 - 28001 Madrid Tels. (91) 276 22 08/09

Programación

TODO SOBRE READ Y DATA

- COMO FUNCIONAN READ Y DATA
- LA SENTENCIA RESTORE
- COMO CONSTRUIR GRAFICOS SENCILLOS A PARTIR DE LOS DATA

Haciendo que el ordenador lea por medio de la sentencia READ un gran volumen de datos te puedes ahorrar tener que teclear programas largos. Puedes usar esta misma técnica para todo, desde gráficos hasta índices sencillos.

Una de las características que hace que un ordenador sea tan versátil es el uso de variables. Y casi siempre la asignación de valores a las variables es algo extremadamente sencillo, no tienes más que escribir, por ejemplo, LET X = 5. Pero hay ocasiones en que la cantidad de información que se quiere utilizar es tan grande que llega a desbordarte. Aquí es donde encuentran su gran utilidad las sentencias DATA y sus acompañantes READ y RESTORE. Observa que estas sentencias no están disponibles en el BASIC del **ZX81**.

La palabra DATA tiene aquí un significado muy específico. Se suele utilizar el término «datos» para designar una gran variedad de cosas. Por ejemplo, un programa, una rutina en código máquina y una matriz almacenada en cinta o en *diskette*, a todo esto se le suele llamar a veces «datos». Pero en todo este artículo al hablar de datos, nos estamos refiriendo a las sentencias DATA y a los elementos que contienen.

El primer método que se suele aprender para asignar valores a una variable, ya se trate de una variable numérica o de una cadena de caracteres, suele ser por medio de una sentencia INPUT. Pero ésta sólo resulta de utilidad cuando es el propio usuario el que suministra la información al ordenador cada vez que se ejecuta un programa.

Muchas veces sin embargo, hay valores fijos que no tienen que ser introducidos ni modificados por el operador, por lo que se pueden incorporar permanentemente en el programa. Este es naturalmente el caso típico de aplicación de la sentencia LET. Pero cuando el volumen de información es muy grande, el uso de grandes masas de sentencias LET es muy tedioso de programar y además la ejecución del programa resulta bastante lenta.

Aquí tienes un ejemplo en el que se podría utilizar una lista fija de encabezamientos en un programa para imprimir un nuevo documento:

```
10 LET A$="DIA"
20 LET B$="SEMANA"
30 LET C$="MES"
40 LET D$="A#O"
50 PRINT A$,B$,C$
   ,D$
```

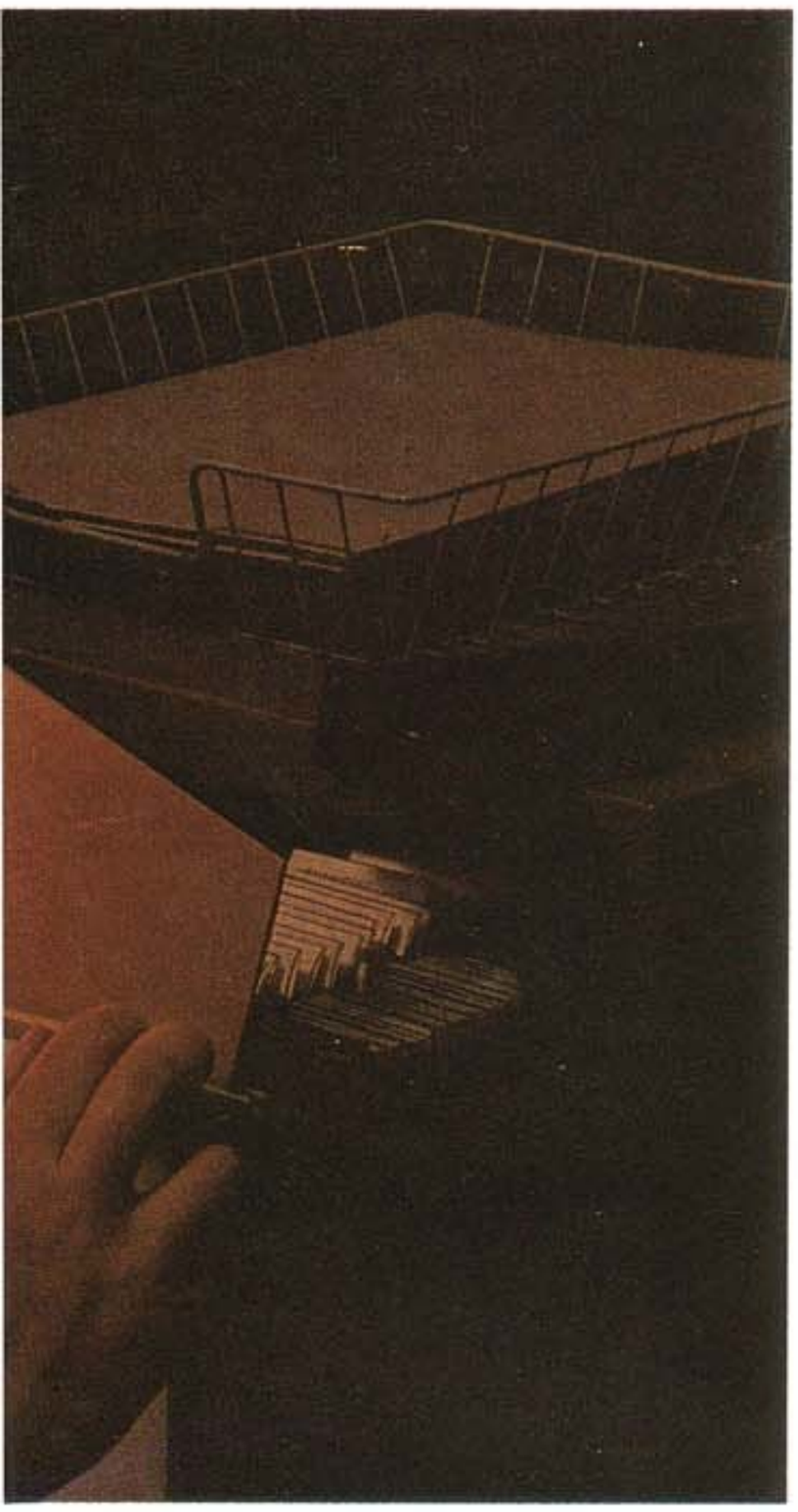
Pero también podrías utilizar sentencias DATA para conseguir el mismo resultado final:

```
10 READ A$,B$,C$,D$
20 PRINT A$.,B$,C$
   ,D$
100 DATA "DIA","SEMANA",
    "MES","A#O"
```

Si ahora resultara que quisieras utilizar no cuatro, sino cincuenta encabezamientos diferentes, con el primer programa se requerirían 46 sentencias adicionales, mientras que con el segundo sólo harían falta una o dos más, dependiendo del ordenador y del tamaño de los encabezamientos.

COMO FUNCIONAN LAS SENTENCIAS «READ» Y «DATA»

¿Cómo funcionan realmente las sentencias READ y DATA? Cuando



el ordenador se encuentra con una instrucción READ, explora todo el programa hasta que encuentra la primera sentencia DATA. A continuación asigna el valor del primer dato de dicha sentencia a la variable que figura en la sentencia READ.

Así, en el programa anterior la cadena de caracteres «DIA» queda asignada a la variable A$, a continuación se asigna «SEMANA» a la variable B$, etc.

Las sentencias DATA aparecen normalmente al final de los programas para «no estorbar» mientras el resto del programa se encuentra en fase de escritura y depuración. Pero realmente pueden ir situadas en cualquier parte. El ordenador simplemente ignora todas las sentencias DATA a menos que una sentencia READ le ordene hacer lo contrario. El siguiente programa funcionaría perfectamente:

```
10 DATA ALEMANIA
20 READ A$,B$
30 DATA "FRANCIA","ITALIA",
   "ESPA#A"
40 READ C$,D$
```

Sin embargo es evidente que el programa resultará mucho más fácil de leer si todas las sentencias DATA se ponen agrupadas en alguna parte. Además hay una regla que hay que respetar siempre: Los datos de las sentencias DATA han de aparecer en el orden en que el ordenador deba leerlos.

El número de elementos que puede contener una sola lista de datos depende de la máxima longitud de línea del ordenador. En cuanto una línea esté llena se empieza una nueva; el ordenador sigue tomándolas en orden.

DIFERENTES TIPOS DE «DATA»

Hasta ahora hemos estado hablando de sentencias DATA que contenían cadenas de caracteres, pero también pueden contener números. El **Spectrum** te permite además utilizar variables y funciones, por lo que puedes perfectamente tener una línea como la siguiente:

```
DATA "PARIS",256,a*5
```

Las variables que aparezcan en la sentencia READ tienen que aparecer en el mismo orden que los elementos de la sentencia DATA. Por esta razón, la línea DATA anterior deberá ser leída con la siguiente sentencia:

```
READ A$,N,X
```

Como puedes observar, la primera variable es una cadena de caracteres, lo que se corresponde con el primer dato que figura en la sentencia DATA. Sin embargo los dos datos siguientes son variables numéricas, debido a que tienen que leer números.

PROBLEMAS QUE PRESENTAN LAS LINEAS «DATA»

Es fácil cometer errores al introducir los datos de una sentencia DATA. Los principales problemas se presentan cuando no tienes suficientes elementos, o cuando intentas que la sentencia READ lea datos de tipo equivocado. Si la sentencia DATA del ejemplo anterior hubiera sido tecleada erróneamente de la forma DATA 256,PARIS,a*5 te encontrarías con que la variable A$ sería '256'. Evidentemente es un resultado erróneo que tu **Spectrum** se encargará de rechazar.

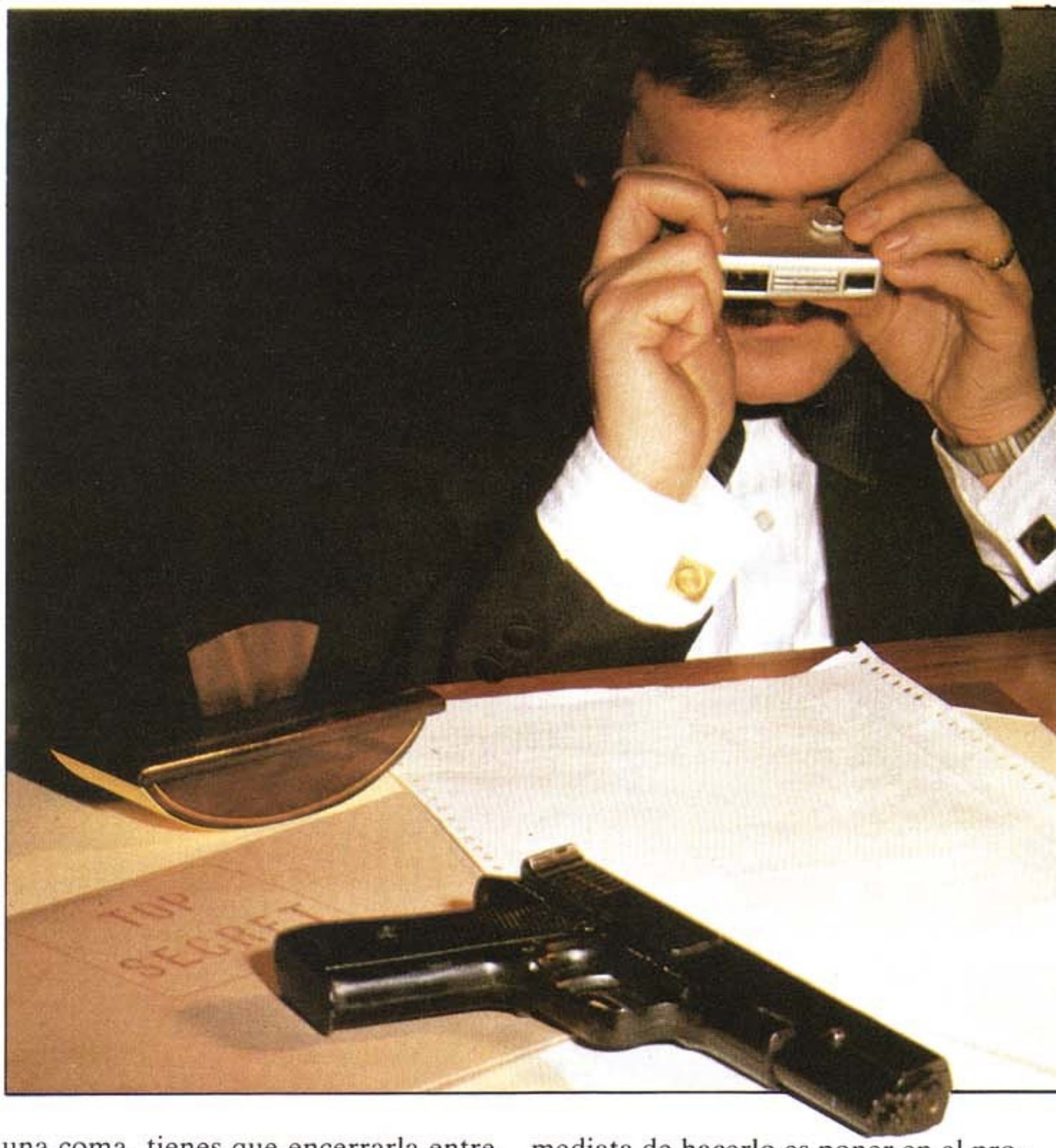
Otro caso distinto se plantea cuando el ordenador intenta leer mediante la sentencia READ la variable 'PARIS', asignándoselo a la variable N. Lo normal es que ocurran una o dos cosas. El ordenador podría darse cuenta que el tipo de los datos está equivocado, enviando un mensaje de error tal como «*type mismatch*» o «mal los datos». Pero también pudiera ocurrir que el ordenador suponga que PARIS es un nombre de variable con el que no se había encontrado hasta ahora, con lo cual te contestaría algo así como «variable no encontrada»

El otro error más frecuente es suministrar datos de menos. Esto ocurre muchas veces cuando estás utilizando un bucle que contiene la sentencia READ, como ocurre en el programa que veremos a continuación. En cuanto haya un ligero error en el parámetro del bucle, o se deje fuera accidentalmente algún dato, el ordenador intentará leer más allá del final de la lista de datos. El resultado es que tu ordenador se detiene, enviándote un mensaje de error que dice «faltan datos» («*Out of Data*»).

UNA LISTA SENCILLA

Aquí tienes un programa que utiliza un bucle para leer los datos con las sentencias READ y DATA. Es un programa muy sencillo de lista de teléfonos. Tú introduces el nombre de la persona y el ordenador te presenta su número de teléfono.

```
5 CLS
10 PRINT AT 2,6;"DIRECTORIO
   TELEFONICO"
20 INPUT"ESCRIBE EL NOMBRE";
   R$
30 FOR J=1 TO 5
40 READ N$,T$
50 IF N$=R$ THEN PRINT AT 10
   ,1"EL NUMERO DE ";N$;" ES"
   ;T$:STOP
60 IF N$="FIN" THEN PRINT AT
   10,3;R$;" NO ESTA EN LA
   LISTA"
70 NEXT J
500 DATA "PEPE","425 12 34",
    "LUIS","876 23 45",
    "ARTURO","567 34 56",
    "RAUL","784 34 12","FIN"
    ,"FIN"
```



Teclea el programa, pero ten la precaución de utilizar en la línea DATA los nombres y números de teléfono de tus amigos. Puedes poner todos los datos que quieras. Lo único que tienes que cuidar al final es de ajustar correctamente el contador de bucle de la línea 30.

Observa que la lista de datos se termina con FIN,FIN. De esta forma la línea 60 puede comprobar si se ha alcanzado el final de la línea. Si el programa ha podido leer hasta aquí, significa que no ha encontrado el nombre, por lo que imprime un mensaje para decírtelo. Hay que introducir «FIN» dos veces, debido a que la línea 40 lee los datos de dos en dos, con lo que de no ser así obtendrás un mensaje de error de falta de datos.

Fíjate también en que para leer un número de teléfono se considera a éste como una cadena de caracteres. Se ha hecho así para respetar un signo de puntuación, el guión, que evidentemente no puede figurar como un número. Las cadenas de caracteres que figuren dentro de una sentencia DATA pueden contener espacios, pero no pueden contener comas. Como los diferentes datos están separados por comas, el ordenador consideraría la coma como el final de uno de los datos. Si te vieras obligado a introducir en una sentencia DATA una cadena de caracteres que contenga una coma, tienes que encerrarla entre comillas. Sería necesario hacer esto en una sentencia DATA como la que figura en el ejemplo siguiente:

```
10 READ N$,A$,B$,C$
20 DATA "LUIS PELAEZ","C/
   PINTO, 25","MURCIA"
```

USO DE «RESTORE»

Con los métodos que llevamos explicados hasta este momento, un programa es capaz de leer mediante una sentencia READ la correspondiente lista de datos de la sentencia DATA, pero sólo lo hará una vez a menos que se vuelva a ejecutar de nuevo el programa. Si quieres que se vuelvan a leer los datos, debes incorporar en tu programa una sentencia RESTORE.

Supongamos por ejemplo que quieres mirar los números de teléfono de algunos de tus amigos. La forma inmediata de hacerlo es poner en el programa una rutina de «Otra Vez». Pon primero STOP en la línea 50 junto con GOTO 80, y a continuación añade las siguientes líneas de programa:

```
80 PRINT AT 12,0;"QUIERES
   OTRO NUMERO (S/N)?"
90 PAUSE 0
100 IF INKEY$="S"THEN GOTO 5
110 IF INKEY$="N"THEN GOTO
    2000
```

¿Qué sucede cuando intentas ejecutar el programa? La primera vez el programa funciona perfectamente. Pero cuando pulsas una tecla para intentar ejecutarlo por segunda vez, te falla porque le faltan datos. Ello se debe a que durante la primera ejecución llegó hasta el final de la lista de datos. Afortunadamente hay una solución muy sencilla para este problema. Teclea:

15 RESTORE

Esta vez el programa funciona bien todas las veces. Ello se debe a que la instrucción RESTORE hace que el ordenador vuelva hasta el principio de la lista de datos.

Es una buena idea poner RESTORE cerca del principio del programa durante la fase de desarrollo. Esto te evita tener que recorrer todas las listas de datos en las pruebas de ejecución. Si crees que más adelante te va a convenir suprimir dicha línea de RESTORE, pon una sentencia REM con un mensaje para que te lo recuerde.

Por medio de RESTORE, puedes volver a utilizar una lista de datos tantas veces como sea necesario. Esto resulta particularmente útil en programas que tienen etiquetas, encabezamientos, tablas, etc, que serán repetidas en momentos diferentes. Por ejemplo, puedes poner los meses del año en una lista de datos dentro de una sentencia DATA en un programa de calendario del tipo que sea, el cual sin duda hará un repetido uso de dichas tablas.

La instrucción RESTORE es especialmente útil en los casos en los que deseas seguir leyendo con READ una lista para buscar un determinado elemento, como es el caso del programa de lista telefónica que hemos visto anteriormente.

USO DE LA SENTENCIA «DATA»

Las listas de datos que contienen las sentencias DATA son extremadamente útiles en toda clase de programas. Puede que hayas hecho uso de ellas en los programas de gráficos definidos por el usuario y en el dibujo del laberinto. También hemos visto cómo puede leer el ordenador los parámetros de la música y otros efectos sonoros.

En los juegos de aventuras, se suele utilizar con frecuencia gran cantidad de líneas de datos en las que se pone todo el texto necesario. Los juegos de marcianitos escritos en BASIC también usan con frecuencia las sentencias DATA para definir personajes y otros elementos del juego.

Los programadores más expertos utilizan las sentencias DATA para hacer programas en código máquina o en lenguaje ensamblador. Tales programas pueden consistir en un corto bucle que contiene comandos POKE los cuales leen (mediante READ) los datos de las sentencias DATA.

En una palabra, todo programa que contenga texto ordinario, números o funciones, puede hacer uso con gran provecho de las listas de datos. Con un uso cuidadoso estas listas pueden convertirse en una poderosa herramienta de programación.

USO DE LOS PUNTEROS DE «RESTORE»

El único problema que presentan las sentencias DATA es que siempre hay que llamar a la información que contienen en la misma sucesión, empezando por la primera posición. Con el uso de RESTORE siempre puedes volver al principio de la lista, incluso aunque todavía no hayas llegado hasta el final. ¿Pero cómo podrías hacer para saltar al centro de una lista?

El **Spectrum** te brinda una manera de hacer esto. En vez de utilizar únicamente una lista, puedes usar realmente varias, haciendo que el ordenador se dirija a la lista apropiada. En el programa de gráficos que figura más adelante, prueba a introducir las siguientes líneas adicionales:

```
6 INPUT "PUENTE VIEJO O
  MODERNO";Y$
7 IF Y$="V"THEN RESTORE 1000
8 IF Y$="M"THEN RESTORE 2000
9 IF Y$<>"V" AND Y$<>"M"
  THEN GOTO 5
2000 DATA 83,84,85,86,87,88,
     89,90,90,90,90,90,90,90
     ,90,89,88,87,86,85,84,
     83
2010 DATA 42,55,63,65,63,55,
     42
```

Los números que figuran a continuación de los comandos RESTORE se llaman punteros de RESTORE. Dirigen al ordenador hacia una determinada lista de datos que empieza en el correspondiente número de línea. Según esto, si pulsas V para tener una casa vieja, el puntero de RESTORE toma el valor 1000.

De hecho, cuando quieras que el ordenador se dirija al primer elemento de la lista DATA, no hace falta utilizar un puntero. Si se hace un RESTORE que no va acompañado por ningún número de línea, equivale a un RESTORE con el número de la primera línea DATA. La línea 7 del anterior programa del **Spectrum** podría haberse escrito igualmente como IF Y$=«V» THEN RESTORE, obteniendo exactamente el mismo resultado.

Los punteros de RESTORE son muy útiles en la programación de juegos. Por ejemplo en un juego de aterrizajes, podrías escribir un programa para comprobar si has aterrizado con suavidad o te has estrellado. Las listas de datos podrían contener en ese caso

SIGUIENDOLE LA PISTA A TUS DATOS

Sea lo que sea lo que contengan los datos de tus sentencias DATA, todos ellos suelen tener una cosa en común: se requiere bastante tiempo y trabajo para calcularlos y teclearlos. Esto se aplica principalmente a la primera vez que trabajas con el programa, pero puede que también resulte necesario volver sobre el mismo mucho más adelante.

Cuando las sentencias DATA se usan para definir un bloque gráfico o algo por el estilo, organiza las líneas del programa para que se correspondan directamente con las filas del gráfico. Si los datos siguen un formato que se repite (como en los números de una lista de teléfonos, por ejemplo), organiza las líneas de programa con todas las entradas de la misma forma.

información para emitir el correspondiente ruido de impacto o una fanfarria victoriosa, y el puntero de RESTORE apuntaría en cada caso a la lista correcta.

USO DE LOS «DATA» EN LOS GRAFICOS

Las sentencias DATA son muy útiles en los programas de gráficos para fijar las coordenadas que hay que dibujar y para llamar a las rutinas gráficas de la ROM. El uso de los DATA de esta manera resulta ideal para las formas irregulares como las que figuran en la siguiente página, ya que se necesitarían muchas líneas de programa si sólo se emplearan instrucciones individuales de impresión o dibujo.

No obstante existen casos en que una sentencia DATA no es de gran utilidad. Así ocurre cuando estás dibujando formas muy regulares en las que las coordenadas o el contorno gráfico pueden calcularse fácilmente.

Las sentencias DATA del programa que veremos a continuación, han sido divididas deliberadamente en varias líneas. Se ha hecho así porque la posterior modificación de los datos puede ser difícil a menos que se tengan unas divisiones muy claras; imagínate lo que sería volver sobre este programa dentro de unos meses e intentar localizar y modificar unos cuantos datos de cada una de las variables. Naturalmente, puedes seguir detenidamente la pista de cada una de las instrucciones READ, pero hacer esto en un programa largo es muy tedioso. Por ello te recomendamos que fragmentes cada grupo de sentencias DATA en varias líneas y que además (suponiendo que tengas memoria disponible) utilices abundantemente las sentencias REM para que las cosas te resulten más claras.

El siguiente programa utiliza READ y DATA para ayudarte a dibujar un puente. Si introduces y ejecutas el programa por etapas, resultará más fácil comprender lo que sucede y comprobar que no te has equivocado al teclear. Empieza con lo siguiente:

```
10 FOR T=74 TO 80 STEP 3
20 PLOT 35,T
30 DRAW 175,0,-2.5
40 NEXT T
```

Esta sección utiliza un bucle FOR ... NEXT que permite el dibujo de tres puntos próximos al lado izquierdo de la pantalla y a continuación dibuja una línea curva a partir de cada uno de dichos puntos. En la línea 30, habrás reconocido que 175,0 significa «hasta un punto situado 175 *pixels* a la derecha del *pixel* de partida». El valor de −2.5 hace que dicha línea sea un arco de circunferencia en lugar de ser recta.

```
100 FOR N=18 TO 39
110 READ A
120 PLOT N,45
130 DRAW 0,A
132 PLOT N+188,45
134 DRAW 0,A
140 NEXT N
1000 DATA 70,70,67,67,70,70,
     60,60,57,57,60,60,57,57
     ,60,60,70,70,67,67,70,
     70
```

Esta sección es la que dibuja las torres. El bucle situado entre las líneas 100 y 140, más el número 45 (pixels contados a partir del fondo) de las líneas 120 y 132 dibuja los puntos de la base de cada una de las líneas verticales muy próximas que forman las torres.

Seguidamente entran en acción las líneas 110 y 1000. La línea 110 dice al ordenador que lea en la línea 1000 la altura, expresada en *pixels*, de cada una de las 22 líneas verticales. Así, la primera línea es 0,70 es decir, vertical y de una altura de 70 *pixels*; la segunda línea es 0,70, la tercera 0,67 y así sucesivamente. Si quieres ver cómo va sucediendo esto, ensaya a insertar temporalmente una instrucción como 135 PAUSE 100 .

```
300 PLOT 0,75
310 DRAW 255,0,-0.1
320 PLOT 0,78
330 DRAW 255,0,-0.1
```

Estas líneas son las que se ocupan de dibujar la carretera; el −0.1 produce una curva ligeramente convexa. Por último:

```
400 FOR R=62 TO 182 STEP 20
410 PLOT R,78
420 READ B
430 DRAW 0,B
440 NEXT R
1010 DATA 42,55,63,65,63,55,
     42
```

Estas líneas se encargan de dibujar los cables verticales que unen el arco con la carretera. El bucle FOR ... NEXT ayuda a dibujar las posiciones de partida de los cables, mientras que las líneas 420 y 1010 controlan su altura.

Si quieres que el programa vuelva a ejecutarse automáticamente, añádele las siguientes líneas:

```
5 CLS:RESTORE
450 GOTO 5
```

La línea 5 borra la pantalla, permitiendo luego que el programa lea de nuevo los datos.

LOS MEJORES DE INPUT SINCLAIR

PUESTO	TITULO	PORCENTAJE
1.º	Commando	17,7 %
2.º	Rambo	14,4 %
3.º	Winter games	10,6 %
4.º	Ping pong	9,9 %
5.º	Skyfox	9,8 %
6.º	Green Beret	9,7 %
7.º	Ole Toro	9,5 %
8.º	Saboteur	7,8 %
9.º	Turbo esprit	5,7 %
10.º	Sir Fred	4,9 %
		100 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «Los Mejores de Input».

Septiembre de 1986.

EL ERIZO HAROLD

Spiky Harold es un divertido programa que, desde el primer momento, nos ha causado una impresión muy favorable. Normalmente, en los juegos de su tipo los gráficos y los protagonistas son de reducidas dimensiones con respecto a la pantalla, llegando en muchos casos a no superar los dos caracteres. Sin embargo, en este caso nos hemos encontrado con un

voluminoso erizo a la búsqueda de provisiones para hibernar, con un tamaño que, a pesar de ser considerable, no resta fluidez a sus movimientos.
Por otra parte, la diversidad de los gráficos, el color, y los desplazamientos de los obstáculos y enemigos que dificultan la tarea de **Harold**, también merecen un comentario muy positivo.
El objeto del juego consiste en conseguir que su protagonista, sin más ayuda que su habilidad para esquivar los muchos peligros a los que ha de enfrentarse, pues aquí no hay láser, ni bombas, ni nada parecido, recobre las fuerzas suficientes para enfrentarse a un largo invierno de letargo. Para ello, deberá comer «todos» los alimentos que encuentre a su paso, y volver a su madriguera antes de que estalle la primera tormenta.
El nivel de dificultad es muy alto, y en algunas pantallas sólo se consigue salir con vidas después de perderlas casi todas, que por cierto, al comenzar son nada menos que veinte.
Después de sufrir el roce mortal de alguna abeja, o una nube de gas, o un inocente pollito, el erizo **Harold** comienza de nuevo en el mismo lugar, hasta perder la última vida. El juego está amenizado por una conocida melodía (El vuelo del moscardón), muy apropiada para poner a prueba vuestra paciencia, sobre todo cuando no podáis salir de alguna pantalla especialmente difícil y comencéis a perder los nervios (esperamos que eso no os ocurra).
Para daros una idea clara del planteamiento del juego, os diremos que se parece un poco al

DATOS GENERALES

TITULO Spiky Harold

FABRICANTE Firebird

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Búsqueda de provisiones

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	6
GRAFICOS	8
COLOR	7
SONIDO	7
TOTAL	35

Profanation, aunque el tema, como habréis podido comprobar, es completamente diferente.
Spiky Harold es un programa estupendo que os recomendamos. En algunos de sus detalles se nota la mano de un programador que gusta de dejar su trabajo bien terminado. Por ejemplo, al realizar un salto, el erizo **Harold** cae «en suspensión», flexionando ligeramente las patas, y sus afiladas púas se mueven apenas perceptiblemente al andar.
En resumen, un buen programa con el que estamos seguros de que pasaréis muy buenos ratos.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

LAS TRES LUCES DE GLAURUNG

«En una oscura y silenciosa sala, perdida en algún lugar del vasto y tenebroso dominio del Señor del Mal...»
Con un comienzo tan literario y sugerente como éste, no cabe la menor duda de que el programa, al menos, tiene un buen argumento. Afortunadamente, también es bueno todo lo demás, empezando por las instrucciones, de las que hemos extraído un párrafo, y pasando por los gráficos, el color y el movimiento.

El objetivo del juego consiste en encontrar tres joyas y una llave, llegar hasta la salida y escapar. Naturalmente, existen multitud de enemigos dispuestos a impedirlo a toda costa, cada uno de ellos con características diferentes. Para

derrotarlos, podrás utilizar algunos objetos que irás encontrando en tu camino, aunque también puedes arriesgarte a un combate directo, que debe variar según el tipo de enemigo al que te enfrentes.
El arma más útili es la flecha, pero de poco sirve contra los dos obstáculos más poderosos del juego: el Mago y el Dragón. LLevando contigo alguna de las tres joyas, podrás acabar fácilmente con ellos. Tanto las joyas como el resto de los objetos, los encontrarás en el interior de los baúles que jalonan el camino, aunque deberás actuar con cuidado, porque algunos de ellos contienen sorpresas muy desagradables.

DATOS GENERALES

TITULO Las tres luces de Glaurung

FABRICANTE Erbe

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Aventura en el castillo

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	7
GRAFICOS	7
COLOR	8
SONIDO	5
TOTAL	34

Las instrucciones son claras y bien estructuradas, y su redacción excelente (esperamos que cunda el ejemplo).
En cuanto a los gráficos y el color, también merecen una valoración positiva, con una mención especial para el profuso colorido de las pantallas. Aunque los fondos son negros, sin apenas decorado, cada personaje y cada objeto es de un color diferente, con lo cual la presentación gráfica es mucho más vistosa que en los programas monócromos.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

EQUINOX

Equinox es una nueva producción de **Mikro-Gen** en la que se abandona, no sabemos si definitivamente, el tema de las aventuras de **Wally**. La ultima vez que tuvimos oportunidad de ver a este simpático personaje fue en **Three Weeks in Paradise** (Tres semanas en el paraíso), programa de aparición relativamente reciente, en el que también estaban **Herbert** (protagonista en solitario del **Dummy Pun**) y el resto de la familia.
Pero en esta ocasión, para desilusión de todos, ya no es el famoso **Wally** quien protagoniza la aventura, sino una esfera-robot teledirigida. No obstante, podemos asegurar desde aquí que no hemos perdido nada en el cambio, pues estamos ante un programa excelente. Y es que a diferencia de otras casas de igual o mayor prestigio, **Mikro-Gen** sigue siendo sinónimo de trabajo bien hecho.

Dicho esto, pasamos a hacer un breve resumen del argumento: El objetivo del juego consite en limpiar de residuos radioactivos una serie de estancias, repartidas en varios niveles, habitadas por un sinnúmero de extraños seres. Para finalizar con éxito la misión contáis con la ayuda de un potente láser, especialmente indicado contra los «bichitos» que tratarán de dificultar el trabajo, y con sofisticados aparatos que os permitirán neutralizar los residuos, pasar de un nivel a otro, y dar el uso correcto a cada uno de los objetos que iréis encontrando.
Aunque se trata de un programa muy original, no esperéis algo

DATOS GENERALES

TITULO Equinox

FABRICANTE Mikro-gen

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Contaminación Nuclear

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	9
GRAFICOS	8
COLOR	7
SONIDO	8
TOTAL	39

radicalmente distinto a las anteriores producciones de esta firma. Se ha seguido un esquema de programación muy similar, con desarrrollo plano en dos dimensiones, y argumento basado en la utilización en el momento y lugar adecuado, de una serie de objetos que se cogen, se usan y se dejan, como en el **Herbert** o el **Pyjamarama**.

Queremos llamar vuestra atención sobre la excelente calidad del sonido, especialmente en la presentación. Nada más cargar el programa, y cada vez que terminamos una partida, podemos escuchar una larga melodía en la que no se repiten sucesivamente breves estribillos, como es habitual, sino que se reproduce nota por nota la partitura de un conocido tema.

Los gráficos son excelentes, el grado de interés muy alto, y la originalidad, con la salvedad que hemos hecho, tampoco deja mucho que desear. En definitiva, **Equinox** es un buen programa al que auguramos un gran éxito.

★★★★★★★★★★★★★★★★★★★★★★★★★★★

LA CIUDAD DE QUAZATRON

Es tan difícil encontrar una pequeña «obra maestra» entre la inmensa avalancha de *soft* de los últimos meses, que ahora que la hemos hallado no podemos dejar de señalarlo, aún desoyendo la regla que dice que «los juicios de valor, se dejan para el final».

TE LO CONTAMOS

Quazatron es un sofisticado programa de **Hewson** de desarrollo tridimensional y cuidadísimos gráficos, que tienen por protagonista al droide de **Meknotech KLP-2**, **Klepto** para los amigos, al que tú tendrás que guiar en su denodada lucha contra los robots que habitan la ciudad subterránea del planeta **Quartech**.

Entre sus muchas y sorprendentes posibilidades, destaca una que es precisamente la que en mayor medida contribuye a dotar al programa de una especial originalidad: no sólo podrás acabar con los peligrosos robots enemigos por el tradicional (y hasta vulgar) sistema del láser, sino que también es posible neutralizarlos desactivando sus circuitos, para después quedarte con aquellos componentes que te sean útiles para reparar los daños de tu **KLP-2**. Es decir, que una vez derrotado el enemigo, se le puede «saquear», pero con cuidado, pues algunos de sus circuitos pueden haber quedado dañados en el combate.

Los robots aparecen identificados por un número del uno al nueve. que nos indica el grado de peligrosidad y sofisticación de cada uno de ellos. Los más difíciles de vencer son los que lucen un bien visible número uno, y los más inútiles los que llevan un nueve.

La ciudad en la que transcurre la acción se divide en varios niveles, accesibles por unos ascensores que

DATOS GENERALES

TITULO Quazatron

FABRICANTE Erbe

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Aventura cibernética

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	9
INTERES	9
GRAFICOS	9
COLOR	6
SONIDO	7
TOTAL	40

los comunican entre si. En algunos puntos encontraréis terminales de ordenador que os permitirán utilizar una serie de valiosas informaciones sobre lo que interesa conocer para lograr el objetivo, como planos tridimensionales, perfiles de los diversos niveles, y una completa biblioteca de datos sobre los componentes electrónicos de los robots enemigos.

Como en todo juego de acción que se precie, el nivel de energía juega un papel muy importante, auqnue en este caso se le ha dado un tratamiento muy original. El rostro de **KLP-2** indica en todo momento, según su gesto, si le falta o no energía. Cuando tenga suficiente, lucirá una amplia sonrisa, y a medida que vayan agotándose las reservas se tornará en un severo gesto de enfado.

LO BUENO Y LO MALO

Como rasgos más novedosos de este sensacional programa, señalaremos su originalísimo planteamiento tridimensional, y su bien logrado efecto de *scroll*, que dan a la imagen una sorprendente sensación de realismo.
El tratamiento de la tercera dimensión en el *soft* para «micros, por lo que al **Spectrum** se refiere, ha pasado hasta ahora por tres etapas: la «prehistoria», con tímidos intentos en los decorados de los fondos; la «revolución» que supuso el **Knight Lore**, con su efecto *Filmation*; la aplicación del *scroll* al escenario de tres dimensiones con el **Higway-Encounter**; y abriendo una nueva etapa, el **Quazatron**, con una original interpretación del espacio y el movimiento.
Poco, por no decir nada, es lo que tiene de malo este programa. La práctica nos está demostrando todos los días que no existe el programa perfecto, pero a veces, programas como éste nos hacen pensar lo contrario.

RESUMIENDO...

Excelente programa.
Desde hace mucho tiempo se viene diciendo que los escasos 48K del **Spectrum** «no dan para más», pero parece ser que hay programas que saben hacer milagros.
En definitiva, se trata de una interesante aventura en la que

además de acción, hay estrategia. No es uno de esos juegos en los que nos matan en cuanto nos descuidamos, y en los que no podemos pararnos a pensar. Acción no falta, pero tampoco sobra.

★★★★★★★★★★★★★★★★★★★★★★★★★★

LA CIUDAD PERDIDA DE SINU

Tres intrépidos exploradores y su fiel amigo **Frozbie** se encuentran en las selvas de América del Sur donde intentarán encontrar los resto de una antigua civilización.
Cada uno de ellos tiene unas características que en un determinado momento pueden facilitar la labor. Así mientras el periodísta **O'Donell** es bastante patoso pero es el más fuerte, el profesor utiliza los objetos hallados en las tumbas, al perro le gusta rascar en la tierra y la bella **Daphne** tiene una especial habilidad para encontrar objetos.
Al inicio de la partida se selecciona el personaje que se desea siendo posible cambiarle durante el desarrollo del juego en fución, las necesidades.
La aventura comienza a las puertas de la ciudad donde un guardián sin cabeza dificulta el acceso a la misma.
Una vez dentro aparecerán las distintas criaturas que intentarán matar a los protagonistas, lógicamente éstos cuentan con unas energías limitadas. Una vez muerto, si alguno de los compañeros que continúan con vida consigue descubrir el brebaje de la inmortalidad podrá resucitarla, y por tanto seguir adelante con la exploración, acudiendo al lugar donde sucumbió.
Se trata de un juego tridimensional en el que los perosnajes se mueven en las cuatro direcciones, pudiendo tomar y dejar objetos, así como grabar el juego y continuarlo después.
En pantalla, además del desarrollo del juego, aparece en la esquina superior izquierda la energía del personaje, en la inferior izquierda el dios del pueblo **Sinu Xipe Totec**, a continuación una ventana con el menú correspondiente y el procentaje del recorrido, mientras alrededor de la pantalla figuran diversos objetos de la cultura **Sinu**.
Cada personaje puede transportar tres objetos simultáneamente aunque no todos los personajes pueden coger todos los objetos.
También existe la posibilidad de

DATOS GENERALES

TITULO Pyracurse

FABRICANTE Hewson

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Civilización perdida

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	6
INTERES	6
GRAFICOS	7
COLOR	5
SONIDO	8
TOTAL	32

llevar conjuntamente a otros personajes por si son necesarios en una misma habitación o zona, pero ello implica un mayor riesgo ya que pueden ser atacados más fácilmente.

En resumen, un entretenido juego de **Hewson** con agradable música y buenos gráficos y color.

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

MISION EN EL PLANETA ZY8OR

Con tu codiciada «Licencia de combate» de la **Federación Planetaria** en el bolsillo, partes hacia el planeta **Zybor** en tu primera misión oficial. Precisamente, la misma misión en la que perecieron tus predecesores.

Zybor no es una simple guarida de piratas estelares, como tantas hay y tantas destruye la Federación, sino un mundo habitado por una raza superior de humanoides, cuya única misión es proteger a toda costa a los cuatro criminales más peligrosos de toda la galaxia. Ellos son nada menos que **Xtro II**, **Ariel Head**, **Max Porka** y **Yokohama**. Todos los que intentaron acabar con alguno de ellos, murieron sin remisión. Pues bien, a ti corresponde intentarlo una vez más, sólo que no bastará con matar a uno, sino que tendrás que mandar a peor vida a los cuatro.

El escenario en el que transcurre la acción es tridimensional, con proyección isométrica, y nos recuerda un poco al **Quazatron**, aunque no llega a la calidad gráfica de ese magnífico programa.

Existen once tipos distintos de objetos: cuatro de ellos son humanoides asesinos, y el resto pulsadores (que son la principal fuente de energía vital de los cuatro criminales), cubos de fuerza, y objetos de diversos usos, como joyas, dinero, maquinaria, etc.

El grado de interés, la calidad de los gráficos y el movimiento, están a un nivel medio-alto con relación a los programas de su tipo, y en general sólo hemos encontrado un par de aspectos negativos: el sonido, muy pobre; y el *scroll*, algo tosco. Por lo demás, podemos decir que se trata de un buen programa.

DATOS GENERALES

TITULO Mantronix

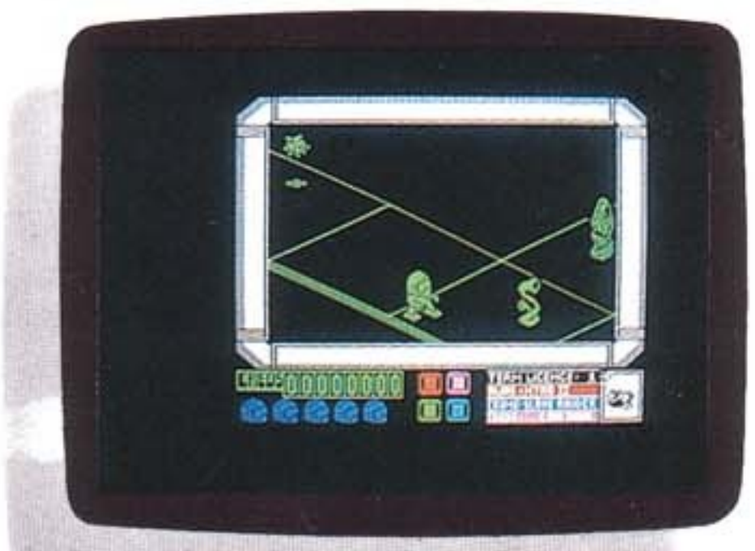
FABRICANTE Probe Software

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Aventura Espacial

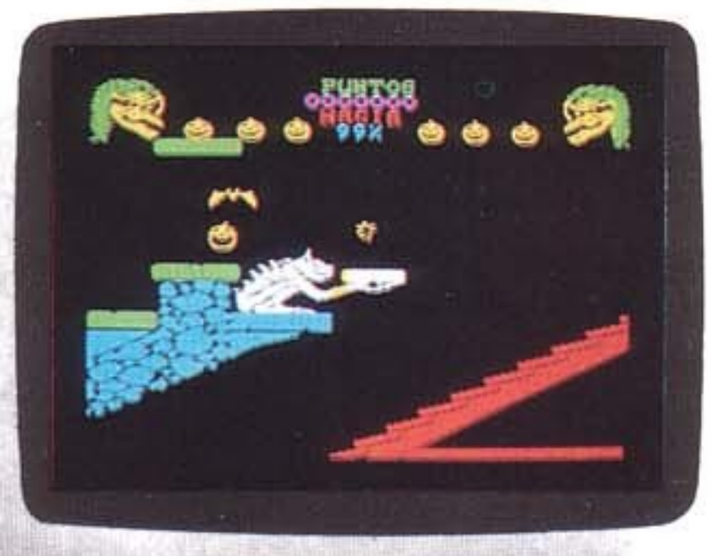
CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	8
INTERES	6
GRAFICOS	7
COLOR	4
SONIDO	4
TOTAL	29



LA CALABAZA CONTRAATACA

Después del éxito del **Cauldron I**, **Palace Software** acaba de lanzar una segunda parte que no tiene nada que envidiar a la primera.
En esta ocasión sí que podemos hablar de una verdadera «continuación» y no de una «repetición». Tanto los gráficos como el planteamiento del juego, han sido renovados hasta el punto de que si hubiera sido presentado el programa con otro nombre, difícilmente podría encontrarse su relación con el **Cauldron I**. El único punto que pone ambas partes en contacto es el tema: también va esta vez de brujas, magia , escobas y calabazas, aunque con un tratamiento diferente. En fin, que si has jugado con la primera parte, puedes comprar sin miedo la segunda (y si no has jugado, naturalmente, también). Por una vez, no se ha puesto en práctica el feo sistema del «autoplagio», que algunos califican de «estafa».
Los cambios introducidos, a parte de una completa renovación del repertorio gráfico, son los siguientes: La bruja, que en la primera parte hacía las veces de protagonista, ahora es la «mala» y tiene un papel secundario. La heroína es esta vez una calabaza, que a falta de extremidades debe desplazarse a saltos, calculando el impulso y la distancia necesarios para llegar hasta donde quiere (al principio, cuesta acostumbrarse). En lugar de los amplios «exteriores» del anterior escenario, tenemos un tenebroso castillo-laberinto, con salones, cuevas y mazmorras. En cuanto al objetivo del juego, los cambios han sido pocos: una vez más, se trata de reunir los ingredientes de una fórmula, con la ayuda de las rimas que van dando «pistas» sobre lo que se debe hacer para lograrlo.



DATOS GENERALES

TITULO Cauldron II

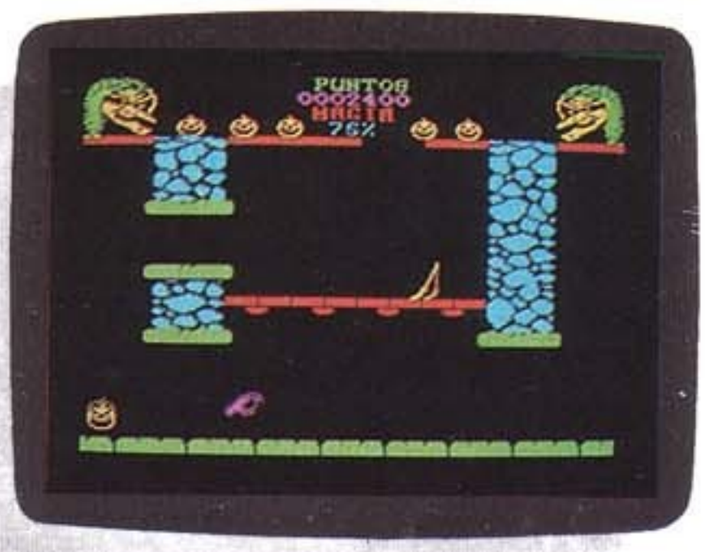
FABRICANTE Palace Software

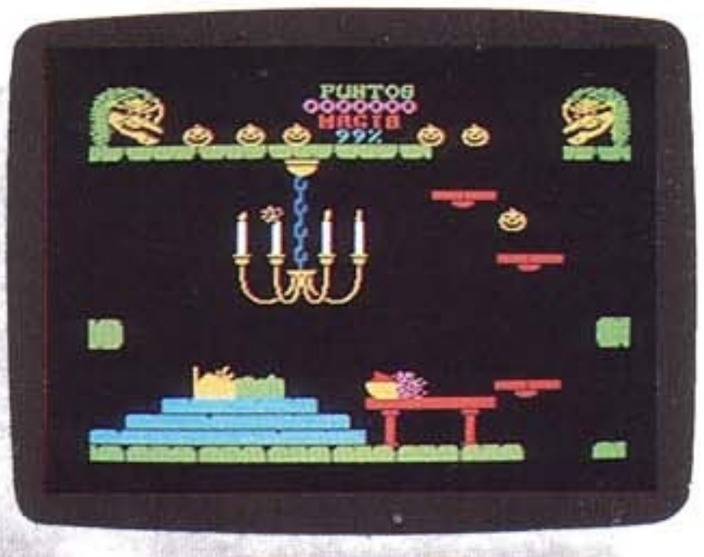
ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Magia y Hechizos

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	8
GRAFICOS	8
COLOR	8
SONIDO	7
TOTAL	38





★★★★★★★★★★★★★★★★★★★★★★★★★★★★

LAS ANDANZAS DE UN ANDROIDE

Un planeta acaba de emerger de un agujero negro y si su nucleo no es reconstruido rápidamente explotará, lo que implica la destrucción de todo el universo.
En una lejana galaxia un androide que ha sido programado durante varios años, está preparado para comenzar su peligrosa misión.
El juego se inicia en la parte más alta de un complicado laberinto que encierra un sin fin de sorpresas.
Existe un número indeterminado de seres que tratan por todos los medios de restarle energías, para lo cual le buscan incansablemente ya que con su simple roce le causan serios daños.

Para llevar a cabo la misión el andoride puede moverse a izquierda y derecha, además de ascender y disparar, aunque estas dos opciones están limitadas, por lo que es necesario saber dosificarlas.
A lo largo del recorrido existe una serie de plataformas que contienen una pequeña nave, la cual permite el transporte en todas las direcciones y por supuesto con una gran rapidez.
El único inconveniente es que mientras el androide utiliza la plataforma volante, no puede recoger ninguno de los objetos que surgen en su camino, por lo que es necesario dejar la nave y continuar a pie, para conseguirlos.
La parte superior de la pantalla muestra, además de la puntuación, el número de vidas que restan y el estado actual de las energías, el rayo láser y el elevador.
A pesar de que los niveles disminuyan, en el laberinto existen bastantes objetos que pemiten renovar esos niveles para continuar la aventura.
Un juego lleno de colorido y acción creado por **Bubbles Bus** que cuenta además con una entretenida banda sonora.

DATOS GENERALES

TITULO Starquake

FABRICANTE Bubble Bus

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Laberinto espacial

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	7
GRAFICOS	8
COLOR	8
SONIDO	8
TOTAL	38

★★★★★★★★★★★★★★★★★★★★★★★★★★★★

MAESTRO DE KUNG-FU

La firma **Ocean** recibió el encargo, por parte de **U.S. Gold**, de hacer una adaptación para **Spectrum** del famoso programa **Kung-Fu Master**, con el que tantas veces hemos jugado en las máquinas tragaperras. Pues bien, el resultado no ha podido ser mejor.

Una vez más, el hecho de que el *hardware* del **Spectrum** carezca de *sprites* se ha dejado sentir. Esto, unido a la imposibilidad de dar color en «alta resolución», ha determinado que los gráficos se quedaran a «años luz» de la versión original: las figuras son monócromas, y sus colores se mezclan al superponerse.
No obstante, el repertorio de golpes, fintas y movimientos, la variedad de los personajes, la emoción y el interés, se han conservado intactos.
También se ha sacado el máximo partido a las pobres posibilidades sonoras del **Spectrum**, incorporando una música de «suspense» muy apropiada para el tema.
Seguro que ya conocéis el argumento, pero por si no es así, os

DATOS GENERALES

TITULO Kung-Fu Master

FABRICANTE U.S. Gold

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA
Artes Marciales

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	7
INTERES	9
GRAFICOS	7
COLOR	6
SONIDO	8
TOTAL	37

lo resumimos:
Tu, el héroe, debes rescatar a la joven cautiva en el quinto piso de la guarida del **Mago**. En cada nivel, deberás derrotar a tus enemigos y llegar a las escaleras del nivel siguiente, antes de que se te acaben las fuerzas o se agote tu tiempo. En el transcurso de la aventura, te enfrentarás a dragones, serpientes, sicarios, globos místicos, avispas, gnomos, guardianes y guerrilleros.
Una posibilidad interesante que incorpora el juego es la de poder elegir el nivel en el que queremos comenzar a jugar. Al principio, parece que esto se refiere al «nivel de dificultad», pero en realidad se trata de los pisos de la guarida. Es decir, eligiendo el nivel cinco, comenzamos en el quinto piso.
Las instrucciones (al menos en el ejemplar que hemos probado), están en castellano. Son bastante completas e incluyen gráficos que ilustran los diversos movimientos que puede realizar el héroe.

En la carátula se advierte que es necesario usar *joystick*, y además en plural (literalmente: «*joysticks* necesarios»). No sabemos si se trata de un error de traducción, o se refiere a las versiones para otros modelos de ordenador, pero la verdad es que no se puede conectar dos *joysticks* simultáneamente, ni tampoco es necesario poseer uno para poder jugar (también se puede manejar el programa desde el teclado).
Para terminar, tenemos que decir que **Kung-Fu Master** es un excelente programa de artes marciales, que podemos incluir entre los mejores de su género.

****

VECTRON

Te encuentras en el interior de un intrincado laberinto en una dimensión desconocida. Viajando a velocidad vertiginosa, debes destruir todas las máquinas asesinas que en él habitan y encontrar después una salida para escapar.
Dispones de un detallado mapa que podrás consultar en cualquier momento, aunque sin detenerte, en el que aparecen las posiciones que ocupan tus enemigos y la tuya propia. Cada color, corresponde a un tipo distinto de máquina.
Vectron es una versión mejorada del tradicional laberinto «de aristas», definido por una serie de líneas que forman planos sin rellenar de color.
La principal innovación introducida es la posibilidad de superponer un mapa transparente sobre la pantalla, que permite ver lo que está ocurriendo y a la vez ser consultado.
El nivel de dificultad es alto, y no existe manera alguna de reducir la velocidad del juego. El punto de mira, que se emplea tanto para disparar como para hacer los giros, se mueve también con excesiva rapidez, y basta un ligero toque a la tecla correspondiente para que se vaya de un lado a otro de la pantalla. Al principio, cuesta un poco acostumbrarse.
La velocidad con la que ocurre todo en el juego, es un elemento muy positivo desde el punto de vista técnico, y el hecho de que incremente mucho el nivel de dificultad tampoco es algo negativo, pero un programa incómodo de manejar nunca es deseable.
No podemos decir que no sea un buen programa, pero la verdad es que tampoco tiene muchos aspectos positivos que reseñar.
Naturalmente, seguro que a los amantes de este tipo de juegos, les va a gustar.

DATOS GENERALES

TITULO Vectron

FABRICANTE Firebird

ORDENADOR Spectrum 48K

TEMA DEL PROGRAMA Laberinto

CALIFICACION (Sobre 10 ptos.)

ORIGINALIDAD	6
INTERES	6
GRAFICOS	5
COLOR	4
SONIDO	4
TOTAL	25

EL ZOCO

Desearía entrar en contacto con usuarios del Spectrum Plus, para intercambios de ideas, trucos e información. También vendo las cintas: Chess, Reversi, Backgammon, De Psion. Interesados escribir a:

Germán Crespo
Buril, 159
Tel. (954) 52 42 36
41006 Sevilla

Vendo cintas y revistas Microhobby Cassette, Microhobby Semanal y Video Basic muy baratas. Me gustaría también cambiar ideas y juegos con usuarios del Spectrum en Andalucía.

Carlos Ramírez Pérez
Placeta del Comino, 7
Tel. (958) 22 72 91
18010 Granada

Vendo consola de juegos Atari con cuatro mandos (dos joysticks y dos paddles), cuatro juegos muy buenos, todos los cables e instrucciones de uso por 25.000 ptas., todo en muy buen estado y funcionando, también lo cambiaria por Spectrum. Preferentemente compradores de la provincia. Llamar a:

Juan
Tel. 437 04 68
Barcelona

Vendo ZX Spectrum 48K RAM, nuevo, junto con alimentador de 9 V., cables de conexión, manuales, tanto en inglés como en español, cintas de juegos y de aplicaciones prácticas, con o sin grabadora, por la cantidad de 28.000 ptas. (negociables).

Juan
Tel. (981) 25 98 95
La Coruña

Urge. Vendo ZX Spectrum 48K, cables y todo, manual en castellano más joystick-interface junto con buenos juegos y utilidades, por 25.000 ptas. Discutibles. Interesados llamar a:

José o Manolo
Tel. (91) 675 16 80

Busco programa de base de datos, lo cambio o lo compro. Cambio programas para ZX Spectrum. Interesados dirigirse a:

José Hernández
Pompeo Fabra, 1
Igualada (Barcelona)

Urge comprar Commodore 64 en buenas condiciones por un máximo de 25.000 ptas, sin datacassette. También vendo ordenador SVI-328 + Shot II + programas + documentación. Todo por 40.000 ptas.

Julio Molina Robledo
Constitución, 11, 7°A
Tel. (911) 22 34 55
19003 Guadalajara

Vendo Commodore Plus/4 a estrenar, con transformador, cables de conexión y dos manuales: uno del Basic 3.5 y otro de 4 programas (Procesador de texto, Base de datos, Gráficos y Hoja de Trabajo) que vienen con el ordenador. Todo por 41.000 ptas, discutibles.

Juan Carlos Martínez
Larrunarri, 5, 3°B
Tel. (943) 52 20 40 (noches)
Rentería (Guipuzcoa)

Vendo copiador inglés turbo, alta velocidad, con o sin cabecera y programas de un sólo bloque por mil quinientas pesetas. Llamar a:

Tel. (94) 415 91 97 (a partir de las seis)
De lunes a jueves

Desearía intercambiar programas de Softwarek, ideas, etc. con usuarios de Spectrum 48K.

Jordi Farnós Vives
San José, 106, 1°
Tel. (93) 803 08 59
Igualada (Barcelona)

Vendo Spectrum ZX 16K en buen estado. Lo dejaré barato. Además regalo cinco cintas con programas. Vendo también cinco libros para el Spectrum. Interesados dirigirse a:

Jesús María Garrués Alonso
Esteban Armendariz, 7
Villava (Navarra)

Urge por servicio militar. Vendo ZX Spectrum Plus (menos de dos meses) con instrucciones (inglés y castellano), cinta Horizonte, transformador y accesorios. Todo por 30.000 ptas. (no negociables). Incluye grabadora.

Javier
Tel. (91) 633 09 37
Madrid

Compro interface 1 + 10 incluso 2 microdrives. Llamar a:

Iñigo Losa
Tel. (943) 39 48 71

Me gustaría contactar con usuarios de Spectrum para intercambio de ideas y opiniones en torno al Spectrum. Soy principiante.

Alberto Echevarría Callejo
Helguera de Reocín, 85
Tel. (942) 82 03 72
Cantabria

Intercambio programas para Spectrum 48K, con gente de Gijón. Poseo juegos, últimas novedades, como son el Sir Fred, Dam Busters, Winter Games, Evil Crown, Beach Head II, Comando, Basketball International, Gyroscope, Glass, Bounty Bob, etc.

Julio César Villa
Felicidad, 4, 4°A
Tel. 39 84 76 (llamar de 14,30 a 15,30
y de 18,15 a 21,30)
Gijón (Asturias)

Cambio libro «ZX Spectrum, qué es, para que sirve y cómo se usa» de Tim Langdell, por juegos preferentemente de guerra.

Juan J. Albiac
Hernández Pacheco, 1, 1°H
10002 Cáceres

Vendo ZX Spectrum con: Televisor de 14 pulgadas Grundig con cassette y radio incorporadas. (B/N), Teclado profesional Lo-Profile (importación), Interface Joystick tipo Kempston y Joystick Quick Shot 2, Curso de Basic con 20 cassettes y 20 libritos (Video Basic), 65 revistas Microhobby, 6 revistas Input Sinclair, 2 libros de programación en Basic, y más de 60 juegos comerciales (buenos). Todo con menos de un año de uso. Precio 80.000 ptas. Preguntar por:

Miguel
Tel. 350 53 36 (llamar preferentemente
de 14 a 15 h.)
Barcelona

Desearía intercambiar ideas, programas, etc., con usuarios del Spectrum. Responderé a todas las cartas.

Javier Zubeldía
Avda. de Madrid, 7, 8°A
San Sebastián (Guipuzcoa)

Compro Interface para Microdrive y/o Microdrive que estén en buenas condiciones y a precio asequible.

David Mark Baber
Venus, 9, 8°C
Tel. (91) 619 56 16
S.J. de Valderas (Madrid)

ASES DEL AIRE

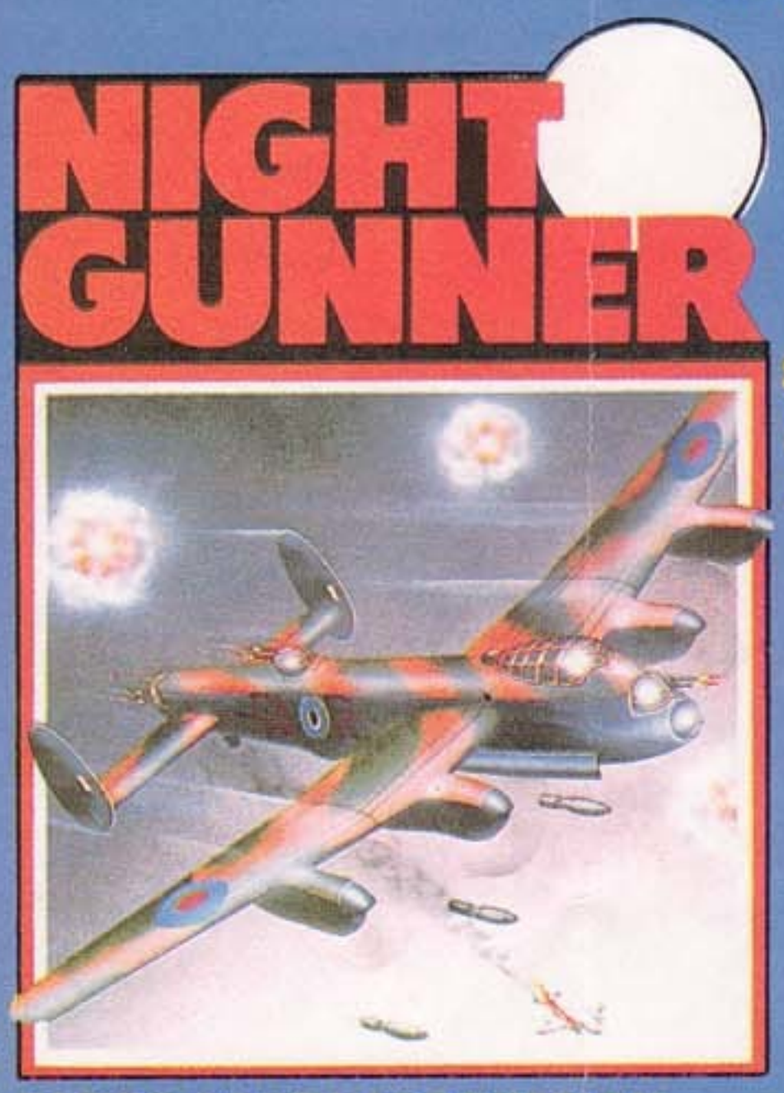
ZAFI CHIP

En las batallas aéreas sólo los mejores sobreviven.

TOMAHAWK

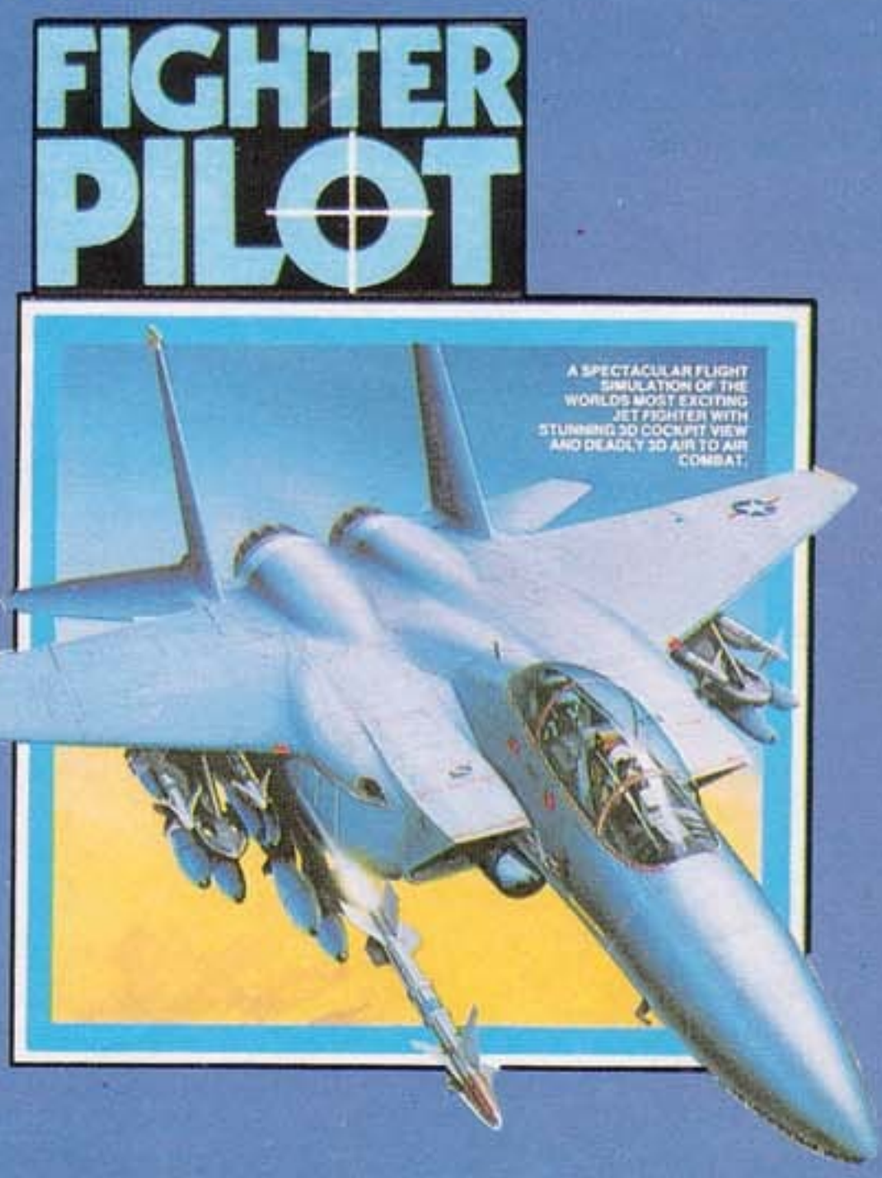
Amstrad Spectrum Amstrad Disk

NIGHT GUNNER



Spectrum
Amstrad
Amstrad Disk

FIGHTER PILOT



Spectrum
Commodore
Amstrad
Amstrad Disk

ZAFIRO SOFTWARE DIVISION
Paseo de la Castellana, 141. 28046 Madrid
Tel 459 30 04. Tel. Barna. 209 33 65. Télex: 22690 ZAFIR E

¿LO HUBIERA PODIDO COMPRAR MAS BARATO?

Los clientes de Regisa esta pregunta ya no se la hacen. Pero además cuando conozcan las **nuevas ofertas** de monitores, ordenadores, impresoras, unidades de disco, periféricos, software, etc. **(evidentemente todo con garantía),** que ha preparado Regisa, se van a llevar una agradable sorpresa.

ventas al mayor

REGISA

Comercio, 11 - Tel. 319 93 08 - Barcelona

lo mismo y más..., pero al mejor precio.

sinclair AMSTRAD SPECTRAVIDEO SEIKOSHA DK-TRONIC

C= commodore HIT BIT SONY :RITEMAN: FONTEC

MONTERO & ASOCIADOS

Establecimientos recomendados: • BAZAR DELHI. Reina Cristina, 11. Barcelona • INTERJOYA. Reina Cristina, 9. Barcelona • BAZAR TAIWAN. Plaza Palacio, 19 (Galerías). Barcelona • LOS GUERRILLEROS. I. Canarias, 128. Valencia • BAZAR KARDIS. I. Canarias, 130. Valencia • BAZAR DELHI. M. Ruano, 5. Lleida